Chapter 2

POLYHEDRAL THEORY AND BRANCH-AND-CUT ALGORITHMS FOR THE SYMMETRIC TSP

Denis Naddef
Laboratoire Informatique et Distribution, Institut National Polytechnique de Grenoble, France
Denis.Naddef@imag.fr

1. Introduction

In this chapter we deal with the problem of solving symmetric TSP (STSP) instances to optimality. Of course, STSP instances are particular cases of asymmetric TSP (ATSP) instances, those for which the distance between any two cities is irrelevant of the direction. Therefore we could transform any instance of the STSP to an asymmetric one and use the results of Chapter 4 to solve it. In fact the techniques of Chapter 4 do not perform well when the costs of the arcs (i, j) and (j, i) only slightly differ. Progress in the solution techniques for the STSP is such that it is common to transform an ATSP into a symmetric one to solve it to optimality (see [474]).

Various methods have been proposed, among which we can cite, for historical reasons, a Lagrangian relaxation algorithm based on *1-trees* by Held and Karp (see [444] and [445]) and a dynamic programming approach that can be found in any text book dealing with dynamic programming. Both methods are very limited, even today, in the size of problems they can solve to optimality. Belloni and Lucena have revisited the Lagrangian approach to the TSP in [98] and report promising results at least for instances that are not too large. The only method that has given good, not to say impressive, results is the Branch-and-Cut method using the double index formulation of the problem. This method finds its foundations in the seminal work of Dantzig, Fulkerson and Johnson all the way back in 1954 [239].

G. Gutin et al. (eds.), *The Traveling Salesman Problem and Its Variations*

The next section contains the double index integer linear programming model we will use. We will also mention other possible formulations of the problem as integer linear programs. We will explain why a substantial amount of theoretical work has to be carried out in order to solve large instances of the problem. Those theoretical developments will be the subject of sections 3 to 12. The following three sections will be devoted to the use of these developments to solve TSP instances to optimality. From now on, in this chapter, *solve* will be understood as *solve to optimality*, which includes a proof of optimality, even if this proof is not easily checkable. Numerical results will be given to show that the method indeed works very well on a large set of instances.

We assume the reader knows the basic concepts of linear algebra dealing with polyhedra.

Except when specified, we assume that we deal with a complete graph $K_n = (V, E)$, with $n = |V|$ representing the number of vertices. Let $S \subset V$, then $\delta(S)$ (resp. $\gamma(S)$) represents the set of edges with exactly one endnode in S (resp. both endnodes in S), i.e. $\delta(S) = \{(u, v) \in E : u \in S, v \notin S\}$ (resp.$\gamma(S) = \{(u, v) \in E : u \in S, v \in S\}$). The edge set $\delta(S)$ is in general called the *coboundary* of S (some authors say *cocycle* of S or *cut* defined by S). We write $\delta(v)$ instead of $\delta(\{v\})$ for $v \in V$. For $S \subset V$ and $T \subset V \setminus S$ we denote by $(S : T) = (T : S)$ the set of edges with one endnode in S and the other in T. We denote by $\mathbb{R}^E$ the set of vectors indexed by E, that is the components of a vector of $\mathbb{R}^E$ are in one to one correspondence with the elements of E. For $E^\star \subset E$ and $x \in \mathbb{R}^E$ we let $x(E^\star)$ represent $\sum_{e \in E^\star} x_e$. Let $G(S)$ denote the induced subgraph on S, i.e. $G(S) = (S, \gamma(S))$. Let $x^\star \in \mathbb{R}^E$, with $x^\star_e$ viewed as the *capacity* of edge e; for $S \subset V$ we call $x^\star(\delta(S))$, the *value* of the cut defined by S. In the following we will abusively say cut S for cut defined by S, that is for $\delta(S)$.

The STSP instances we will mention are taken from the TSPLIB [709]. The number in the name of the instance represents the number of cities.

2. Integer linear programming models

There are various ways of modelling the STSP as an integer linear program. Some contain a polynomial (in n) number of variables and constraints, they are said to be *compact*, some others only a polynomial number of variables but an exponential number of constraints.

Since a STSP instance can be transformed into an asymmetric one by replacing each edge by two oppositely directed arcs of same cost, all the integer formulations for the ATSP yield formulations for the STSP. There are various known compact formulations for the ATSP, see for

example [650] and [291]. We will mention two compact formulations for the STSP that do not come from the ATSP.

For computational reasons, which will become clear in this chapter, among two integer models, the best is the one whose linear relaxation has the highest objective function value. In order for this to make sense one must assume that the integer linear programming model is minimal in terms of inequalities, since any integer linear programming model can be strengthened by the use of valid inequalities for the underlying polyhedron, which will be the subject of most of this chapter. This strengthening, at least theoretically, could achieve a value equal to the optimal integral value.

Among all the integer linear programming models known, in this respect, three models emerge and attain the same value for their linear relaxations. None is known that attains a higher value. These are the *multistage insertion* formulation of Arthanari [39] (see also [41]), the *cycle shrink* of Carr [169], and finally what is known as the *double index* or *subtour elimination* formulation. This latter formulation will be extensively described here since it is the only one that has been used so far in computational studies. The multistage-insertion is inspired by dynamic programming recursion, that is building up a tour step by step. Cycle shrink does the opposite, that is going from a tour to a node. It is not surprising then that these two formulations are equivalent, see Arthanari and Usha [42] (see also [40] for the equivalence with the double index formulation). For the ATSP, Padberg and Sung [650] show that the double index formulation dominates all the other known ones.

We now describe the *double index* formulation. To every edge $e \in E$ we associate a variable x_e which will take value 1 if e is in the resulting optimal tour and 0 else. The STSP can then be formulated as the following integer linear program:

$$\text{(IP(STSP))} \qquad \min \sum_{e \in E} c_e x_e \tag{1}$$

subject to

$$x(\delta(v)) = 2 \quad \text{for} \quad v \in V \tag{2}$$

$$x(\delta(S)) \geq 2 \quad \text{for} \quad 3 \leq |S| \leq |V|/2 \tag{3}$$

$$0 \leq x_e \leq 1 \quad \text{for} \quad e \in E \tag{4}$$

$$x_e \text{ integer} \quad \text{for} \quad e \in E \tag{5}$$

Equations (2) just say that two edges have to be chosen incident to any vertex. Inequalities (3) forbid cycles which do not contain all the vertices and are called *subtour elimination inequalities*. If $S = \{v\}$, i.e. $|S| = 1$, the corresponding Inequality (3) is implied by the Equation (2)

corresponding to v. Since $\delta(S) = \delta(V \setminus S)$, we can restrict the subtour elimination inequalities to sets S with at most half the vertices. The number of subtour elimination inequalities is still in $\Omega(2^n)$, that is, exponential in the size of the problem. This may seem to be a serious problem, we will discuss this point later on.

The name *double index* formulation comes from the ATSP in which both extremities of the arcs play different roles and therefore x_{ij} is used instead of x_e, creating two indices. When dealing with the STSP this is not the case, so we stick to variables x_e and will avoid using the misleading terminology "double index" and rather use the other usual name of *subtour elimination* formulation which comes from the Inequalities (3).

Why deal with an exponentially large integer model while the other two equivalent, in terms of strength, formulations only have a polynomial number of them? This has to do with the fact that separating these inequalities is not a problem. We will make this concept of separation more precise shortly.

Why do we prefer formulations with high optimal linear relaxation value? This has to do with the solution technique which goes as follows.

The linear relaxation $LP(STSP)$ of $IP(STSP)$ is the linear program obtained by dropping the integrality conditions (5). The most obvious way to solve $IP(STSP)$ is via Branch-and-Bound using as lower bound the value of the linear relaxation of the subproblems (see for example [209], [826]). The method goes as follows (assuming we are minimizing, which is our case) and that the term feasible relates to the *integer* linear program:

A Branch-and-Bound algorithm for integer linear programming

- **Initialization:** The set of subproblems to solve is initialized to the linear program obtained by dropping the integrality conditions on the variables. This problem is referred to as the root subproblem.
- **Choose a subproblem:** If the list of open problems is empty, the best known feasible solution is optimal. Else choose an open subproblem and delete it from the list.
- **Treat subproblem:** Solve the associated linear program. If the solution is integer, go back to *Choose a subproblem* after eventually updating the best known integer solution and the best known solution value.
- If the value of the objective function exceeds that of the best known feasible solution, go back to *Choose a subproblem.*

- Else, using some linear inequality, partition the current subproblem into two new subproblems. The union of the feasible (integer) solutions to each of these two subproblems contains all the feasible solutions of the problem that has been partitioned. This is commonly done by choosing a variable with a current fractional value $\overline{x}_e$, and by creating two subproblems which are added to the list of open problems, one in which we impose $x_e \geq 1$ and another in which we impose $x_e \leq 0$. (There are other ways of doing this, which will be exposed in due time.) These two subproblems are called the *sons* of the current subproblem. Go to *Choose a subproblem.*

The reader may realize that, as described here, the algorithm is not of any use for our problem since none of the linear programs could fit on any computer for even relatively small size instances. This is due to the number of constraints which is exponential in the number of cities of the TSP instance. One way of getting around this is known as *constraint* or *row generation*. One call to the solution of a linear program, in the previous algorithm, will now be replaced by a series of calls. In our case it goes as follows.

Assume we have an algorithm that for a given vector $\overline{x} : E \to \mathbb{R}$, seen as capacities on the edges of G, returns a *minimum capacity cut* of G. Such an algorithm returns a set $S \subset V$ such that $\overline{x}(\delta(S))$ is minimum. The Branch-and-Bound algorithm described above is modified in the initialization and the treatment of each subproblem as follows:

- **Initialization:** The set of subproblems to solve is initialized to the linear program consisting of Constraints (2) and (4).

- **Treat subproblem:** Until the subproblem is not declared examined, repeat: Solve the current linear program which yields the optimal solution $\overline{x}$. Search for $S \subset V$ such that $\overline{x}(\delta(S)) < 2$. If no such S exists, the subproblem is **examined**, else, add $x(\delta(S)) \geq 2$ to the set of constraints of the current linear program.

When one has finished treating the first subproblem, which we refer to as the *rootnode problem*, one has *optimized over the STSP subtour elimination polytope*.

We are now concerned with two questions:

- Is there an efficient algorithm to find a minimum capacity cut in a weighted undirected graph?

- Does the number of calls to the minimum capacity cut algorithm remain reasonable in practice?

To answer the first question, Nagamochi and Ibaraki in [622] and Nagamochi, Ono and Ibaraki in [623] give efficient minimum cut algorithms in the case of undirected graphs. Padberg and Rinaldi in [646] give very efficient preprocessing procedures that significantly accelerate any minimum cut algorithm. One can find a study of performances of several minimum cut algorithms in Jünger, Rinaldi and Thienel [476] and in Chandra, Chekuri, Goldberg, Karger, Levine and Stein [181].

As for the second question, since finding a minimum capacity cut is polynomial, using the ellipsoid algorithm, one can, theoretically, finish in a polynomial number of iterations (see Grötschel, Lovász and Schrijver [399]). If we use the procedure we just described, almost all problems of the TSPLIB require very few iterations.

The value of the linear relaxation obtained from $IP(STSP)$ by dropping the integrality conditions on the variables (i.e. the root problem), is known as the *Held and Karp bound* or *value on the subtour elimination polytope.*

Although the Held and Karp bound is very good, it is not good enough to be able to solve even average-size STSP instances by the procedure just described since the number of subproblems created is far too high. We mention here a conjecture, which seems difficult to trace back to its origin, but that can be found in Goemans [383] .

Conjecture 1 *If the edge costs satisfy the triangular inequality, then the optimal value of a tour is at most* 4/3 *the value of the Held and Karp bound.*

Consider the partition of the vertex set V into V_i^j, $i = 0, \ldots, k$, $j = 1, \ldots, 3$, with $V_0^j = A$ and $V_k^j = Z$ for $j = 1, \ldots, 3$, and $k \geq 3$. In Section 6 this will be called a $(k-1)$- path configuration. Let $c_e = 0$ if $e \in \gamma(V_i^j)$ for all i and j, $c_{uv} = 1$ if $u \in V_i^j$ and $v \in V_{i+1}^j$ for all i and j, $c_{uv} = k-2$ if $u \in A$ and $v \in Z$. For all the other edges $e = (u, v)$, let c_{uv} equal the length of a shortest path linking u to v using only edges for which the cost has been defined in the preceding lines. For such an instance of STSP, the optimal value of a tour is $4k-2$ and the Held and Karp bound is $3k$, therefore the ratio can be made as close as one desires from the bound of the conjecture.

For all the examples of the TSPLIB, the gap is much less than the one we would expect from the conjecture.

The more general procedure known as Branch-and-Cut differs from the algorithm described to optimize on the subtour elimination polytope in the way a subproblem is treated. One not only tries to generate constraints coming from the integer linear formulation but also any linear inequality that separates the current fractional solution from the feasible

solutions. Such an inequality is called a *cutting plane* or just a *cut*. Using the term *cut* would be confusing since it is also a set of edges that disconnects an undirected graph. In the following, except for the name Branch-and-Cut, we will avoid saying cut for an inequality. Remember that we will use cut S in place of cut $\delta(S)$.

The next sections will show the existence of a finite set of linear inequalities, which can be added to $IP(STSP)$ in such a way that the linear relaxation of the newly obtained integer program is always integral and therefore corresponds to an optimal tour. Unfortunately, as we will see, this is only an existence theorem and it is unlikely that we will discover that complete set of linear inequalities in a near future even for medium size instances. Table 3.1, which will be commented in the next section, gives the number of such inequalities for instances up to 10 nodes. Fortunately, even a partial knowledge of that description is of precious help in solving STSP instances, and that will be the topic of the next sections.

In view of all this the **Treat subproblem** phase of a Branch-and-Cut looks like this:

Treat subproblem. Repeat until a stopping criterion is attained:

- Solve the current lp, let $\overline{x}$ be its fractional optimal solution $\overline{x}$.
- Find a linear inequality $fx \geq f_0$ satisfied by all incidence vectors (see definition in next section) of tours and such that $f\overline{x} < f_0$.

Two stopping criteria are given in Section 19.4.

3. Introducing the symmetric TSP polytope and its various relaxations

We let $\mathcal{H}_n$ (resp. $\mathcal{H}_G$) represent the set of tours of the complete graph K_n (resp. of graph G). To every tour $\Gamma \in \mathcal{H}_n$ (resp. $\in \mathcal{H}_G$), we associate a vector $x^\Gamma \in \mathbb{R}^E$ such that $x_e^\Gamma = 1$ if $e \in \Gamma$ and $x_e^\Gamma = 0$, else. This vector is called indifferently *incidence* or *representative* vector of Γ. Some authors use *characteristic* vector of Γ.

3.1. The symmetric TSP polytope

The *symmetric TSP polytope* of G, *STSPP(G)*, is the convex hull of all the vectors x^Γ when Γ ranges over all Hamiltonian cycles of G. When $G = K_n$, we let $STSPP(n)$ stand for $STSPP(K_n)$, i.e. $STSPP(n) = conv\{x^\Gamma : \Gamma \in \mathcal{H}_n\}$. When not specified, we assume the underlying graph is K_n.

n	# tours	# different facets	# facet classes
3	1	0	0
4	3	3	1
5	12	20	2
6	60	100	4
7	360	3 437	6
8	2 520	194 187	24
9	20 160	42 104 442	192
10	181 440	$\geq$ 51 043 900 866	$\geq$ 15 379

Table 2.1. Some statistics on $STSPP(n)$

It is well known from a theorem of Weyl [823], that $STSPP(n)$ can be described by a *finite* set of linear equations and inequalities. The study of the symmetric TSP Polytope consists in finding such linear equations and inequalities. All the linear *equations* are known in the case of $STSPP(n)$, this is the basis of the proof of Theorem 2. This is not the case for $STSPP(G)$ if G is an arbitrary connected graph.

Before going further on we give an idea of the richness of the facial structure of STSPP(n). Table 3.1 is taken from Christof and Reinelt [186]. This table gives, for small values of n, the number of tours, the number of different facets and the number of classes in which they can be put. Two facet inducing inequalities for $STSPP(n)$ are said to belong to the same class if a renumbering of the vertices transforms one into the other. One can observe that this polytope is highly degenerate, that is the number of facets incident to a vertex greatly exceeds the minimum number of facets needed to define that vertex. As can be seen from this table, the complexity of the polytope increases very fast and so does its degeneracy. The numbers in the last line are conjectured to be exact.

The first thing to do, when studying a polyhedron, is to determine its dimension. That dimension is not known for $STSPP(G)$ when G is a connected arbitrary graph. Note that $STSPP(G)$ is non empty if and only if G is Hamiltonian.

For complete graphs, the polytope,dimension is given by the following theorem.

Theorem 2 *The dimension of* $STSPP(n)$ *is* $|E| - |V|$.

Proof: (sketch): For $n = 3$ there is only one tour, so the theorem is true in that case. Equations (2) are linearly independent, so the dimension cannot be more that what is announced in the theorem. It is therefore enough to exhibit $|E| - |V| + 1$ affinely independent tours. For this we use a theorem on the partition of the edges of K_{n-1}. If $n = 2k + 1$, then the edges of K_{n-1} can be partitioned into k edge disjoint Hamiltonian

cycles. If $n = 2k$, then the edges of K_{n-1} can be partitioned into $k-1$ edge disjoint Hamiltonian cycles and a perfect matching. For n odd, for each of the $(n-1)/2$ disjoint Hamiltonian cycles that partition the edges of K_{n-1} and each edge of that cycle, we create one Hamiltonian cycle of K_n by inserting node n between the endnodes of that edge, i.e. if $e = (i, j)$, we remove edge e and add the two edges (i, n) and (j, n). In the other case, for the Hamiltonian cycles of the partition, we do the same thing. As for the perfect matching, we complete it arbitrarily to a Hamiltonian cycle, and do as previously only with the edges of the perfect matching. We leave it to the reader to check that in both case we have the right number of cycles and that the corresponding vectors are affinely independent. ■

Remark 3 *Another proof not using the decomposition of the edges of K_n is given by Queyranne and Wang in [687].*

Corollary 4 *Any Equation in the description of $STSPP(n)$ is a linear combination of the Equations (2). Therefore Equations (2) are the only necessary equations in a minimal description.*

Remark 5 *In Chapter 11, the problem of characterizing the cost functions such that all Hamiltonian cycles have the same cost (constant TSP) is raised. If one knows a maximal linearly independent set of linear equations $c^i x = c_0^i$ for $i = 1, \ldots p$, satisfied by all the incidence vectors of Hamiltonian cycles, one has the dimension of $STSPP(G)$ and also a solution to the constant STSP problem. Let c be a cost function on the edges such that all Hamiltonian cycles of G have the same cost c_0, then $cx = c_0$ is a linear equations satisfied by all Hamiltonian cycles of G and is therefore a linear combination of the equations $c^i x = c_0^i$ for $i = 1, \ldots p$.*

In the case of complete graphs this yields:

Corollary 6 *(see also Chapter 11) The only cost functions on the edges that have the property that all Hamiltonian cycles of the complete graph have the same cost, are those obtained from any function $\pi : V \to \mathbb{R}$ and by setting $c_e = \pi(i) + \pi(j)$ for all $e = (i, j)$.*

Proof: Let $c : E \to \mathbb{R}$ be such that all the tours have a cost of c_0. Then $cx = c_0$ is an equation satisfied by all tours, and therefore, by Theorem 2, must be a linear combination of the equations (2). The coefficients of that linear combination are the $\pi(i)$'s. ■

Finding the dimension for general graphs is much more complex and only very few results are known. In fact deciding whether $STSPP(G) \neq$

$\emptyset$ is NP-complete. The reason is that the set of equations is difficult to find, since some constraints written as inequalities turn out to be equations. Take for example the case of Figure 2.1. There is an implicit equation, linearly independent from the degree constraints, namely that all tours contain exactly two of the three horizontal edges, therefore the subtour inequality defined by one of the triangles is in fact an equation.

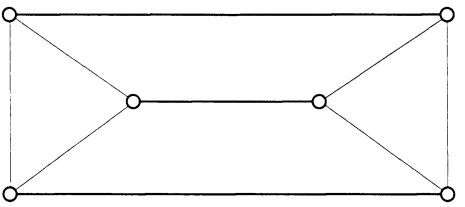

Figure 2.1. The diamond graph

For G a Halin graphs (see Cornuéjols, Naddef and Pulleyblank [220]), a complete description of $STSPP(G)$ is known. This case was generalized to classes of graphs which have the property to "decompose", in some way, using three-edge cutsets (see Cornuéjols, Naddef and Pulleyblank [221]). It would be a nice result to characterize those graphs for which Constraints (2) to (4) of the linear relaxation of $IP(STSP)$ describe $STSPP(G)$. We will mention, later on, such a result but for the convex hull of *spanning closed walks*, a *spanning closed walk* being a cycle visiting each vertex **at least** once (any edge can be used an arbitrary number of times).

From now on we restrict to the case of complete graphs. Non full dimensional polyhedra have the undesirable property that an infinite number of linear inequalities define the same facet, unlike the case of full dimensional ones where a facet is described by a unique inequality up to scaling by a strictly positive factor. The following theorem, which in fact is a corollary of Theorem 2, deals with two representations of the same facet of $STSPP(n)$, where A is the node-edge incidence matrix of K_n.

Theorem 7 *Let $fx \geq f_0$ be a facet inducing inequality of $STSPP(n)$. The valid inequality $gx \geq g_0$ defines the same facet if and only if there exists $\pi : V \rightarrow \mathbb{R}$ and $\pi_0 \in \mathbb{R}$, $\pi_0 > 0$, such that: $g = \pi_0 f + \pi A$ and $g_0 = \pi_0 f_0 + 2\sum_{i \in V} \pi_i$*

Testing whether two facet inducing linear inequalities define or not the same facet can be solved in $O(n^2)$ using the following algorithm of F.Margot [582]. Choose a spanning tree T and add an edge e^* that

creates an odd cycle when added to the edges of T. Solve for the variables π_i, $i \geq 1$ as a function of π_0 using the set of n equations relative to each edge of $T + \{e^*\}$. This can be performed in $O(n)$ time. Solve for π_0 using the right hand sides. In $O(n^2)$ time check whether the solution satisfies the equations relative to all the edges not in $T + \{e^*\}$.

The usual way to tackle the problem of multiple linear descriptions of the same facet, is to embed the non full dimensional polyhedron in a full dimensional one. In general we require the original polyhedron to be a face of the full dimensional one. The larger polyhedron is called a *relaxation* of the smaller one. A first step is to study a minimal linear description of the relaxation, which is unique (up to scaling by a positive integer), and then try to see which facets of the relaxation define facets of the original polyhedron.

We now turn to the various relaxations that have been used to study STSPP(n).

3.2. The monotone Relaxation

Historically this has been the first relaxation used in the study of STSPP(n) [195], [402], [401], [403], [404], [642]. The *monotone STSP polytope* is the convex hull of the incidence vectors of the tours and all edge subsets of tours of K_n. It is trivially of full dimension since the empty set is a subset of a tour, and the sets consisting of a single edge are also subsets of tours, since, in K_n, every edge belongs to some tour (in fact each edge appears in $(n-2)!$ different tours). Balas and Fischetti [74] give some very simple conditions under which facet inducing inequalities of the monotone STSP polytope yield facets of $STSPP(n)$.

We will not detail this relaxation since it seems to have attained its limits and is nowadays only very seldom used. We will pass over relaxations of $STSPP(n)$ using the *2-matching polytope* which has received very little attention (see Cornuéjols and Pulleyblank [224], [225], [223]) and turn to two much more interesting relaxations.

3.3. The Hamiltonian path relaxation

This relaxation is due to Queyranne and Wang [687]. They observed that there is a bijection between the tours of K_{n+1} and the Hamiltonian paths of K_n. Let *HP(n)* denote the convex hull of the representative vectors of all the Hamiltonian paths (no fixed extremities) of K_n. The following theorem states that this polytope is near full dimensional.

Theorem 8 *Dim(HP(n))*$=\frac{n(n+1)}{2} - 1$

Proof: Note that $x(E_n) = n - 1$ holds for all representative vectors of Hamiltonian paths of K_n. Hence the dimension cannot be more than stated. Let $fx = f_0$ be an equation satisfied by all Hamiltonian paths of K_n. Let e_1 and e_2 be any two different edges of K_n and let Γ be a Hamiltonian cycle of K_n containing these two edges, which always exists. Let $P_i = \Gamma \setminus \{e_i\}$, for $i = 1, 2$, be two Hamiltonian paths obtained from Γ by removing one of the edges e_1 and e_2 respectively. We have $fx^{P_1} = f_0$ and $fx^{P_2} = f_0$ and therefore $fx^{P_1} - fx^{P_2} = 0 = f_{e_1} - f_{e_2}$. Therefore $f_{e_1} = f_{e_2}$, and since e_1 and e_2 are arbitrary, f_e is a constant for $e \in E_n$, that is $fx = f_0$ is a multiple of $x(E_n) = n - 1$. ■

The main interest of this approach is that it provides a normalized form for the linear inequalities defining facets of $STSPP(n)$. A facet defining inequality $fx \leq f_0$ for $STSPP(n)$ is in *normalized form* if:

- $f_e = 0$ for all $e \in \delta(\{1\})$
- $f_e \geq 0$ for all $e \in E \setminus \delta(\{1\})$
- $\exists e \in E \setminus \delta(\{1\})$ such that $f_e = 0$
- $\min\{f_e : e \in E \setminus \delta(\{1\}) \text{ and } f_e > 0\} = 1$

Theorem 9 *Every facet defining inequality of $STSPP(n)$ has a unique normalized form*

Proof: See [687]. ■

How difficult is it to find the normalized form of the facet defined by a facet inducing inequality $fx \leq f_0$? Queyranne and Wang give a $O(n^2)$ algorithm to do so. This yields another $O(n^2)$ algorithm to check whether or not two facet defining inequalities define the same facet of $STSPP(n)$. To do so, find the normalized form of the facet defined by each. If they are identical, the facets are the same, else they are different. We will give later on another standard form for $STSPP(n)$ facet defining inequalities, which will only lead to a $O(n^3)$ algorithm for the problem of recognizing whether or not two inequalities define the same facet of $STSPP(n)$.

3.4. The graphical relaxation

Let $G = (V, E)$ be a connected, not necessarily complete, graph. A *spanning closed walk* W of G is a family of edges of G (in a family we assume a same element can appear more than once) such that the graph (V, W^*) is *eulerian*, where W^* is a set of edges obtained from W by replicating an edge as many times as it appears in W. An *eulerian*

graph is a connected multigraph in which each vertex is incident to an even number of edges. Figure 2.2 shows a graph and one of its closed walks. We will **always** assume closed walks to be spanning and therefore omit the word spanning from now on.

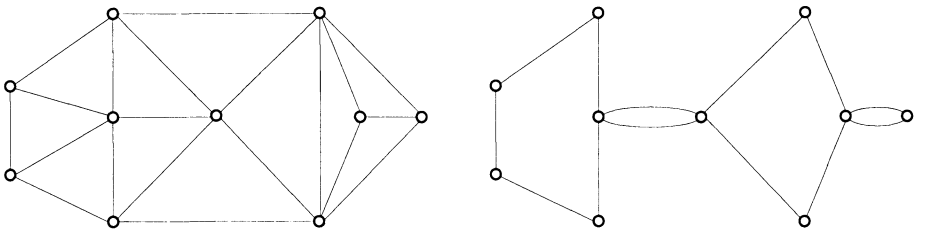

Figure 2.2. A graph and one of its closed walks

We can assign a vector $x^W \in \mathbb{R}^E$ to a closed walk W such that x_e^W represents the number of times e appears in W.

The *Graphical Traveling Salesman Polyhedron of* G, $GTSPP(G)$, is the convex hull of all the representative vectors of closed walks of G. As we will see, $GTSPP(n) = GTSPP(K_n)$ has very strong ties with $STSPP(n)$, which explains why today this is the most used relaxation to study the latter polytope. Note that $STSPP(n)$ is the face of $GTSPP(n)$ defined by $x(E) = n$.

If W is a closed walk of G, then so is $W + (2k)\{e\}$ (the walk obtained from W by going k times back and forth along edge e) for any positive integer k , and therefore $GTSPP(G)$ in an *unbounded* polyhedron if G is connected and has at least two vertices.

The problem of finding a closed walk of minimum cost in G is called the *Graphical Traveling Salesman Problem* on G, $GTSP(G)$. The term "graphical" may not be the most suitable, but at the time it was defined in [219], the idea was to reflect the fact that this version of the traveling salesman problem has an optimal solution as long as the graph is connected. Note that if some edge e is such that $c_e < 0$, then the problem has no finite optimal solution since adding to any solution two copies of edge e strictly decreases its value. If all edge costs are non-negative, then there is an optimal solution in which no edge will appear more than twice. Finally note that if in addition the edge costs satisfy the triangular inequality and $G = K_n$, then there is an optimal solution to $GTSP(n)$ which is also a solution to $STSP(n)$.

Theorem 10 *If G is connected, then $GTSPP(G)$ is of full dimension, otherwise it is empty.*

Proof: If G is not connected, then no closed walk exists. Else let W be any given closed walk. Consider the $|E|$ closed walks $W_e = W + 2\{e\}$ for all $e \in E$. The representative vectors of those $|E|$ closed walks and that of W are affinely independent. ■

The following theorem shows that if G has no *bridge*, that is an edge the removal of which disconnects the graph, then the convex hull of the extreme points of $GTSPP(G)$ is a polytope of full dimension. If edge e is a bridge, then each closed walk which corresponds to an extreme point uses exactly twice that edge, and therefore $x_e = 2$ is an equation satisfied by all such walks.

Theorem 11 *If G is connected with k bridges, then the convex hull of the extreme points of $GTSPP(G)$ is a polytope of dimension $|E| - k$.*

Proof: See [219]. ■

Theorem 12 *The inequality $x_e \geq 0$ defines a facet of $GTSPP(G)$ if and only if the edge e is not a bridge.*

Proof: If e is a bridge, then $x_e \geq 2$ holds for every closed walk of G. If e is not a bridge, then there exists a closed walk W of $G \setminus \{e\}$. Consider the $|E| - 1$ closed walks $W_f = W + 2\{f\}$ for every $f \in E \setminus \{e\}$. The representative vectors of these closed walks together with that of W are affinely independent. ■

Theorem 13 *Let $S \subset V$, then $x(\delta(S)) \geq 2$ is a facet of $GTSPP(G)$ if and only if the induced subgraphs $G(S)$ and $G(V \setminus S)$ are connected.*

Proof: If, say, $G(S)$ is not connected, then $x(\delta(S)) \geq 4$ holds for all closed walks and therefore the inequality $x(\delta(S)) \geq 2$ is not even supporting for $GTSPP(G)$. Conversely, for each $e \in \delta(S)$, let W_e be a closed walk that contains twice e and no other edge of $\delta(S)$. Let W_{e^*} be one of the just defined closed walks. For all $e \notin \delta(S)$, let $W_e = W_{e^*} + 2\{e\}$. The $|E|$ representative vectors of these closed walks are affinely independent and all satisfy the inequality with equality. ■

The reader may be surprised at this point that we did not give a formulation of $GTSP(G)$ as an integer linear program. The problem is that no such formulation is known. All we can write is the following:

$$\min cx \tag{6}$$

subject to

$$x(\delta(S)) \geq 2 \text{ and } \textbf{even} \quad \text{for} \quad 1 \leq |S| \leq |V|/2 \tag{7}$$

$$x_e \geq 0 \text{ and integer} \quad \text{for} \quad e \in E \tag{8}$$

Research Question 14 *Find an integer linear formulation for the graphical traveling salesman problem using only the variables of the double index formulation.*

The previous question needs to be made more precise. It is known that there does not exist a set of linear inequalities with the property that all points with integral components inside the polyhedron defined by these inequalities correspond to closed walks. The question is to find a set of linear inequalities such that all *integral extreme points* of the polyhedron they define correspond to spanning closed walks.

As suggested by M. Fischetti, one could change the space of variables, associating an integer variable $y_u \in \mathbb{N}$ to each node u, and add the constraints $x(\delta(u)) = 2y_u$.

Fonlupt and Naddef [317] characterized those graphs G for which the following set of linear inequalities defines $GTSPP(G)$

$$x(\delta(S)) \geq 2 \quad \text{for} \quad 1 \leq |S| \leq |V|/2 \tag{9}$$

$$x_e \geq 0 \quad \text{for} \quad e \in E \tag{10}$$

A *minor* of a graph G is any graph H that can be obtained from G by recursively performing the following operations in some order.

1 (edge deletion) Remove an edge from the current graph

2 (edge contraction) Remove an edge from the current graph and identify its two extremities.

Theorem 15 *Inequalities (9) and (10) describe $GTSPP(G)$ if and only if the graph G does not contain, as a minor, one of the three graphs shown in Figure 2.3.*

Proof: The proof is very long and technical. See [317]. ■

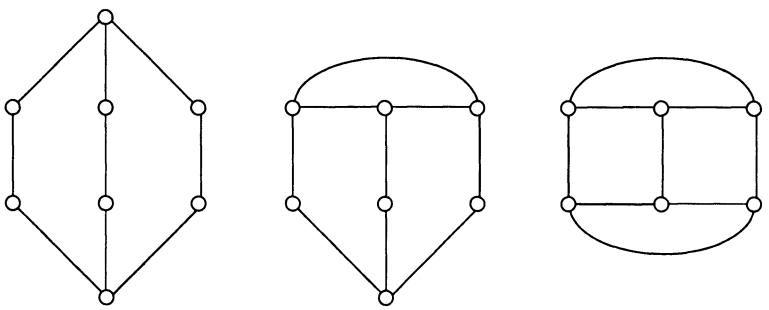

Figure 2.3. Excluded minors for Theorem 15

Graphs without such minors are well characterized, in the sense that they are easy to recognize. There is a small set of *minimal* graphs of the family called *bricks*. Given two graphs of the family, one can compose them in two ways to obtain a new graph of the family. One is by identifying a node of each, the second by identifying two nodes of each. Each non minimal graph of the family is obtained from two smaller ones of the family. Conversely, using disconnecting sets of one or two nodes, we can decompose the graph into smaller graphs recursively. At the end, if all obtained graphs belong to the list of possible starting bricks, then the graph belongs to the family. For more details see [317].

A similar characterization for the convex hull of Hamiltonian cycles is not known. Cornuéjols, Naddef and Pulleyblank, in [220] and [221], give families of graphs, which includes the Halin graphs, for which the convex hull of the Hamiltonian cycles is given by the degree constraints and the subtour elimination inequalities.

There are other known families of TSP instances that can be solved to optimality using only these inequalities in the formulation. But in these families, this is due to the objective function. Papadimitriou and Steiglitz [654] gave a family of TSP instances known as the *Papadimitriou and Steiglitz's traps* designed to defeat any attempt to solve them by a $k-OPT$ procedure, that is a procedure in which at each iteration one tries to replace a tour by a better one differing from it by at most k edges. Padberg and T.Y Sung [649] have shown that these graphs always yield an integral solution to the linear program obtained by Constraints (2), (3) and (4). The polytope on these constraints is in general called the *subtour elimination polytope*. The same has been observed by Althaus and Mehlhorn [20] for TSP problems arising in curve reconstruction. It yields a polynomial reconstruction algorithm. These two last examples are studied in Chapter 11. Moreover, in some experimental results, G. Rinaldi has observed that if the distance between cities is independently and uniformly randomly generated in some large interval, optimizing over the subtour elimination polytope almost always yields integer optimal solutions.

We now study more in depth the strong relationship between $GTSPP(n)$ and $STSPP(n)$.

4. The graphical relaxation Framework

Except when otherwise specified, graphs are assumed to be complete.

4.1. General theorems

We introduce first some notation. The same way that we represented the set of all tours of K_n by $\mathcal{H}_n$, we represent the set of all closed walks of K_n by $\mathcal{W}_n^*$. A *minimal closed walk* is a closed walk that does not contain a subfamily of edges which is also a closed walk. In particular, a minimal closed walk never contains an edge more than twice. The converse is of course false. The set of minimal closed walks is represented by $\mathcal{W}_n$.

From now on we do not differentiate sets of edges and the vectors that represent them. Therefore when we talk of a tour or a closed walk it is either a set of edges or its representative vector.

Given an inequality $fx \geq f_0$, valid for $STSPP(n)$ (resp. $GTSPP(n)$), we call *extremal* those tours (resp. minimal closed walks) which satisfy that inequality with equality. We let $\mathcal{H}_f^=$ (resp. $\mathcal{W}_f^=$) represent the set of extremal tours (resp. minimal walks) with respect to $fx \geq f_0$. We will also often say that a tour of $\mathcal{H}_f^=$ is *tight* for $fx \geq f_0$, the same for the closed walks.

A trivial remark is that for any valid inequality $fx \geq f_0$ of $GTSPP(n)$, we have $f_e \geq 0$ for all $e \in E$. This comes from the fact that if $f_{e^*} < 0$ for some $e^* \in E$, then adding enough copies of e^* to any closed walk will bring its value below f_0, contradicting that it is a valid inequality.

For any inequality $fx \geq f_0$ defined on $\mathbb{R}^E$ and for each node $u \in V$, we define the set $\Delta_f(u)$ by :

$$\Delta_f(u) = \{(v, w) \in E : v \neq u, w \neq u, f_{vw} = f_{uv} + f_{uw}\}. \qquad (11)$$

The set $\Delta_f(u)$ plays a central role in our study of $STSPP(n)$ and $GTSPP(n)$.

Another central notion in the study of $GTSPP(n)$ is that of *tight triangular* inequality.

Definition 16 *An inequality $fx \geq f_0$ defined on $\mathbb{R}^E$ is said to be* tight triangular *or in* tight triangular form *if the following conditions are satisfied:*

(a) The coefficients f_e satisfy the triangular inequality, i.e. $f_{uv} \leq f_{uw} + f_{wv}$ for each triplet $\langle u, v, w \rangle$ of distinct nodes of V.

(b) $\Delta_f(u) \neq \emptyset$ for all $u \in V$

We will abbreviate "tight triangular" by "TT". The following theorem shows that almost all facet defining inequalities for $GTSPP(n)$ are tight triangular.

Theorem 17 *A facet defining inequality $fx \geq f_0$ for $GTSPP(n)$ falls in one of the following three categories:*

(i) trivial inequalities $x_e \geq 0$ *for all* $e \in E$

(ii) degree inequality $x(\delta(v)) \geq 2$ *for all* $v \in V$

(iii) tight triangular inequalities

Proof: Let $fx \geq f_0$ be a facet defining inequality for $GTSPP(n)$. Suppose it does not satisfy condition (a) of Definition 16. Then there is a triplet of vertices u, v, w such that $f_{uv} > f_{uw} + f_{wv}$. If there is a walk $W_{uv} \in \mathcal{W}_f^=$ containing the edge (u, v), then the closed walk $W' = W_{uv} + \{(u, w)\} + \{(w, v)\} - \{(u, v)\}$ is such that $fx^{W'} < f_0$, which contradicts the fact that the inequality is valid. Therefore no closed walk of $W_f^=$ contains the edge (u, v) and the facet is that defined by $x_{uv} \geq 0$. Now assume that the inequality satisfies (a) but not (b) of Definition 16. Therefore there exists $u \in V$ such that for all pairs of distinct nodes v and w of $V \setminus \{u\}$, we have $f_{vw} < f_{uv} + f_{uw}$. If there is a closed walk $W \in \mathcal{W}_f^=$ such that the degree of u in W is at least 4, then there always exists two neighbors t and z (see Figure 2.4) of u on W such that $W' = W + \{(t, z)\} - \{(u, t), (u, z)\}$ is also a closed walk (if (u, t) or (u, z) appears more than once in W, we only remove one copy). We have by hypothesis, that $fx^{W'} < f_0$, contradicting the validity of the inequality. Therefore all closed walks $W \in \mathcal{W}_f^=$ which do satisfy condition (a) and not condition (b) satisfy $x(\delta(u)) = 2$ for some $u \in V$. ■

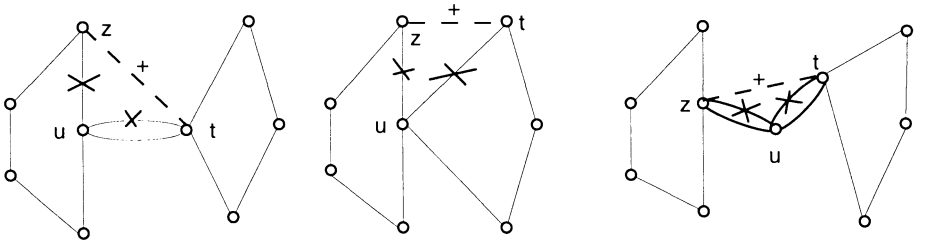

Figure 2.4. Examples of "shortcuts"

Corollary 18 *Let* $fx \geq f_0$ *be a facet defining TT inequality for* $GTSPP(n)$. *Then*

(a) for every edge $e \in E$ *there exists a closed walk* $W \in \mathcal{W}_f^=$ *such that* $e \in W$

(b) for every node $v \in V$ *there exists a closed walk* $W \in \mathcal{W}_f^=$ *such that* v *has degree at least 4 in* W.

Remark 19 *All facet defining TT inequalities $fx \geq f_0$ for GTSPP(n) have the following structural property. There exists a unique partition $V_1, \ldots, V_j, \ldots, V_k$ of the vertex set V such that $f_e = f_{ij} > 0$ for all $e \in (V_i : V_j)$, $i \neq j$, and $f_e = 0$ if and only if $e \in E(V_j)$ for some j, $1 \leq j \leq k$.*

Proof: Let $V_1, \ldots, V_j, \ldots, V_k$ be the vertex sets of the connected components of the graph $G^0 = (V, E^0)$, where $E^0 = \{e \in E : f_e = 0\}$. Let $e \in E(V_j)$ and $W_e \in \mathcal{W}_f^=$ such that $e \in W_e$. Assume $f_e > 0$. There is a path in V_j linking the extremities of e and for which all the edges have a coefficient of 0. The closed walk W^* obtained from W_e by removing e and by adding the edges of that path is such that $fx^{W^*} < f_0$, which contradicts the validity of $fx \geq f_0$. Now let e and e' be two edges of $(V_i : V_j)$ and assume $f_{e'} < f_e$. Let $W_e \in \mathcal{W}_f^=$ such that $e \in W_e$. The closed walk W^* obtained from W_e by removing e and adding the edges of a path in V_i from the extremity of e to the one of e' it contains, the edge e' and the edges of a path in V_j from the extremity of e' to the one of e it contains, does not satisfy the inequality $fx \geq f_0$. Note that the existence of W_e in all cases is guaranteed since otherwise the inequality would be $x_e \geq 0$, which is not tight triangular. ■

If, in the partition of Remark 19, all the sets have cardinality 1, we say that the inequality $fx \geq f_0$ is *simple*. As we will see, it is often convenient to study, first, simple inequalities and then to obtain the more general inequalities via the *node lifting* procedures described in Section 4.3.

Definition 20 *For every ordered triplet $\langle u, t, z \rangle$ of distinct nodes in V, we call* shortcut *on $\langle u, t, z \rangle$ the vector $s_{utz} \in \mathbb{R}^E$ defined by:*

$$s_{utz}(e) = \begin{cases} 1 & \text{if } e = (t, z) \\ -1 & \text{if } e \in \{(u, t), (u, z)\} \\ 0 & \text{otherwise} \end{cases}$$

Let $W \in \mathcal{W}_f^=$ with strictly more than n edges. The following lemma will be very useful:

Lemma 21 *Let $fx \geq f_0$ be a TT inequality supporting for GTSPP(n) and let $W \in \mathcal{W}_f^=$ be a closed walk with $t > n$ edges containing a certain edge e^*. For every vertex u with degree $k \geq 4$ in W, there exists a shortcut s_{uvw} such that $x^W + s_{uvw} \in \mathcal{W}_f^=$, contains e^* and has $t - 1$ edges.*

Proof: Easy, see [618]. ■

Corollary 22 *Let $fx \geq f_0$ be a facet defining TT inequality for $GTSPP(n)$. Then, for each edge $e \in E$, there exists $H \in \mathcal{H}_f^=$ that contains e.*

The following lemma shows how to transform a facet inducing inequality for $STSPP(n)$ into an equivalent one in TT form.

Lemma 23 *Let $hx \geq h_0$ be a facet defining inequality for $STSPP(n)$. An inequality $fx \geq f_0$ equivalent to $hx \geq h_0$ is tight triangular if and only if $f = \pi A + \pi_0 h$ and $f_0 = 2\sum_{u \in W} \pi_u + \pi_0 h_0$ where (π, π_0) satisfy*

$$\pi_u = \frac{1}{2}\pi_0 max\{h(v,w) - h(u,v) - h(u,w) : u,v,w \in V, u \neq v \neq w\} \quad (12)$$

Proof: It is straightforward to check that if (π, π_0) satisfies Equation (12), the inequality $fx \geq f_0$ is tight triangular. Suppose that $fx \geq f_0$ is tight triangular. Condition (a) of the definition of tight triangularity imposes that

$$\pi_u \geq \frac{1}{2}\pi_0 max\{h(v,w) - h(u,v) - h(u,w) : u,v,w \in V, u \neq v \neq w\} \quad (13)$$

The second condition of that definition implies the existence of a triple of distinct vertices u, v, w for which (13) holds with equality. ■

A *basis* of a facet defining inequality $fx \geq f_0$ for $GTSPP(n)$ is a set $\mathcal{B}_f$ of $|E|$ closed walks in $\mathcal{W}_f^=$ whose incidence vectors are linearly independent. A closed walk W is *almost Hamiltonian* in u if u has degree 4 in W and all other vertices have degree 2.

Definition 24 *A basis $\mathcal{B}_f$ of a facet defining inequality $fx \geq f_0$ of $GTSPP(n)$ is called* canonical *if it contains $|E| - n$ Hamiltonian cycles and n almost Hamiltonian walks.*

Theorem 25 *A non-tivial TT inequality $fx \geq f_0$ which is facet defining for $STSPP(n)$ defines a facet of $GTSPP(n)$.*

Proof: We just show how to build a basis $\mathcal{B}_f$. Since the inequality is facet defining for $STSPP(n)$, there exists $|E| - n$ linearly independent Hamiltonian cycles in $\mathcal{H}_f^=$. We now construct n almost Hamiltonian walks, one for each $u \in V$. Let $e = (v,w) \in \Delta_f(u)$ and $\Gamma_e \in \mathcal{H}_f^=$ a Hamiltonian cycle containing e, which exists as the inequality is not equivalent to the trivial inequality $x_e \geq 0$. Let $W_u = \Gamma_e - \{e\} + \{(u,v),(u,w)\}$. All we have to prove is that the previously mentioned $|E| - n$ Hamiltonian cycles together with these n almost Hamiltonian walks are linearly independent. This can be found in [618]. ■

Note that the facet inducing inequality $x_e \leq 1$ for $STSPP(n)$ has a TT-form which is $x(\delta(\{u,v\})) \geq 2$ where $e = (u,v)$, which we already know is facet inducing for $GTSPP(n)$. The following theorem shows the strong relationship between $STSPP(n)$ and $GTSPP(n)$.

Theorem 26 *Every non-tivial facet of $STSPP(n)$ is contained in exactly $n+1$ facets of $GTSPP(n)$, n of which are defined by the degree inequalities.*

Proof: Let F be a non-tivial facet of $STSPP(n)$. Then F belongs to the n facets of $GTSPP(n)$ defined by the degree inequalities $x(\delta(\{v\})) \geq 2$, for all $v \in V$. Every other facet of $GTSPP(n)$ containing F is defined by a TT-inequality. By Lemma 23, there is only one such facet and the theorem follows. ■

This theorem shows that the polyhedral structure of $STSPP(n)$ is very closely related to that of $GTSPP(n)$.

A corollary of this theorem is that the TT-form is a canonical way of expressing the facets of $STSPP(n)$. Therefore, together with Lemma 23, it leads to a $O(n^3)$ algorithm to recognize whether or not two facet inducing inequalities of $STSPP(n)$ define the same facet which, as already mentioned, it is not as good as Margot's or Queyranne and Wang's $O(n^2)$ algorithms.

We will study the polyhedral structure of $GTSPP(n)$ in more detail in the next sections. At this point, the reader may want to skip the rest of this section which addresses very technical aspects of the study of $STSPP(n)$. In particular, we address the problem of finding sufficient conditions for a facet defining inequality of $GTSPP(n)$ to define a facet of $STSPP(n)$. Note that, at the time of writing this chapter, no tight triangular facet inducing inequality for $GTSPP(n)$ is known that is not also one for $STSPP(n)$ (except of course the degree inequalities).

From Definition 24, a facet inducing TT inequality for $GTSPP(n)$ defines a facet for $STSPP(n)$ if and only if it has a canonical basis. In the following we investigate sufficient conditions for the existence of a canonical basis. The following remark is central in the approach to such conditions.

Remark 27 *Let $fx \geq f_0$ be a TT facet defining inequality for $GTSPP(n)$ and $\mathcal{B}_f$ be one of its bases. Let $\{W_u : u \in V\}$ be a set of n almost Hamiltonian walks of $\mathcal{W}_f^=$, where W_u is almost Hamiltonian in u. If every closed walk of $\mathcal{B}_f$ can be reduced to a cycle of $\mathcal{H}_f^=$ by using only shortcuts obtained by a linear combination of the incidence vectors of elements of $\mathcal{A}_f = \mathcal{H}_f^= \cup \{W_u : u \in V\}$, then $\mathcal{A}_f$ contains a canonical basis of $fx \geq f_0$. That is, all the incidence vectors of the closed walks T of $\mathcal{B}_f$ can be expressed as a linear combination of the incidence vectors*

of elements in $\mathcal{A}_f$. In general, it is sufficient that every shortcut can be obtained as a linear combination of the incidence vectors of elements in $\mathcal{A}_f$ with coefficients in $\mathbb{R}$. To obtain simple sufficient conditions we will restrict the coefficients to $\{-1, 0, 1\}$.

Let $e = (u, v)$ and $e' = (w, y)$ be two distinct edges of E. We say that e and e' are *f-adjacent* if there exists a Hamiltonian cycle in $\mathcal{H}_f^=$ containing both e and e'. Let z be a vertex in V, we say that e and e' are *f-adjacent in z* if:

(i) e and e' belong to $\Delta_f(z)$;

(ii) There exists a closed walk $W_z \in \mathcal{W}_f^=$ almost Hamiltonian in z that contains (z, u), (z, v), (z, w) and (z, y);

(iii) $W_z - \{(z, u), (z, v)\} + \{e\}$ is a Hamiltonian cycle (and therefore so is $W_z - \{(z, w), (z, y)\} + \{e'\}$).

A set of edges $J \subset E$ is said to be *f-connected* if for every pair of distinct edges e_1, $e_2 \in J$, there exists a sequence of edges $e'_1 = e_1, e'_2, \ldots, e'_{k-1}, e'_k = e_2$ in J, such that for $i = 1, \ldots, k-1$, e'_i is f-adjacent to e'_{i+1}. A set of edges $J \subset E$ is said to be *f-connected in z* if for every pair of distinct edges, e_1 and $e_2 \in J$, there exists a sequence of edges $e'_1 = e_1, e'_2, \ldots, e'_{k-1}, e'_k = e_2$ in E (not necessarily in J), such that for $i = 1, \ldots, k-1$, e'_i is f-adjacent in z to e'_{i+1}. Observe that the notion of f-connectivity in z is weaker than that of f-connectivity. Any subset of a f-connected set in z is also f-connected in z.

Lemma 28 *Let $fx \geq f_0$ be a TT facet defining inequality of $GTSPP(n)$. If $\Delta_f(u)$ is f-connected in u for every $u \in V$, then $fx \geq f_0$ has a canonical basis, hence it is facet defining for $STSPP(n)$.*

Proof: Let $u \in V$ and $\{W_u : u \in V\}$ be any set of n almost Hamiltonian walks of $\mathcal{W}_f^=$. This set always exists, by Corollary 22, since $fx \geq f_0$ is tight triangular it can be constructed as in the proof of Theorem 25. By Lemma 21 there exists a shortcut $s_{u\overline{yz}}$ with $(\overline{y}, \overline{z}) \in \Delta_f(u)$ that can be used to reduce W_u to a Hamiltonian cycle $\Gamma \in \mathcal{H}_f^=$. We therefore have $s_{u\overline{yz}} = x^\Gamma - x^{W_u}$, hence $s_{u\overline{yz}}$ is a linear combination with coefficients in $\{-1, 1\}$ of incidence vectors of elements from $\mathcal{A}_f = \mathcal{H}_f^= \cup \{W_u : u \in V\}$. If $|\Delta_f(u)| = 1$, we are done. Else let $e = (v, w)$ and $e' = (y, z)$ be two distinct edges of $\Delta_f(u)$ which are f-adjacent in u. Let us assume that the shortcut s_{uyz} is a linear combination with coefficients in $\{-1, 1\}$ of incidence vectors of elements from $\mathcal{A}_f$. Since e and e' are f-adjacent in u, there exist a closed walk $W'_u \in \mathcal{W}_f^=$ almost Hamiltonian in u which contains (u, v), (u, w), (u, y) and (u, z). We have $\Gamma_1 =$

$W'_u - \{(u,y),(u,z)\} + \{(y,z)\} \in \mathcal{H}_f^=$ and $\Gamma_2 = W'_u - \{(u,v),(u,w)\} + \{(v,w)\} \in \mathcal{H}_f^=$, by triangular inequality. The shortcut s_{uvw} can be expressed as: $s_{uvw} = x^{\Gamma_2} - x^{\Gamma_1} + s_{uyz}$, hence it is a linear combination with coefficients in $\{-1,1\}$ of incidence vectors of closed walks from $\mathcal{A}_f$. By the assumption that $\Delta_f(u)$ is f-connected in u, it follows that for all $u \in V$ and all $(v,w) \in \Delta_f(u)$, the shortcuts s_{uvw} can be expressed as a linear combination of the incidence vectors of elements in $\mathcal{A}_f$, and by Remark 27 the lemma follows. ■

The conditions of this lemma are too restrictive in general. We give now a weaker version, which is the one that is in general used in proving facet inducing results for $STSPP(n)$ from results for $GTSPP(n)$.

Lemma 29 *Let $fx \geq f_0$ be a TT facet defining inequality of $GTSPP(n)$. If there exists a basis $\mathcal{B}_f$ of $fx \geq f_0$ and for every $u \in V$ there exists a nonempty set of edges $J_u \subseteq \Delta_f(u)$, f-connected in u, such that every closed walk $W \in \mathcal{B}_f$ can be reduced to a tour of $\mathcal{H}_f^=$ by using only shortcuts from the set $\{s_{uvw} : (v,w) \in J_u, u \in V\}$, then $fx \geq f_0$ has a canonical basis, hence it is facet defining for $STSPP(n)$.*

Proof: For every $u \in V$, let (v,w) be any edge in J_u. By Corollary 22 there exists $\Gamma \in \mathcal{H}_f^=$ containing the edge (v,w). Let W_u be the almost Hamiltonian walk defined by $W_u = \Gamma - (v,w) + (u,v) + (u,w)$. By Remark 27 and a process analogous to that of the proof of the preceding lemma, it follows that the set $\mathcal{A}_f = \mathcal{H}_f^= \cup \{W_u : u \in V\}$ contains a canonical basis for $fx \geq f_0$ and the lemma follows. ■

We now turn to the problem of deriving facet inducing inequalities for $GTSPP(n)$ and for $STSPP(n)$ from other such inequalities. The first way we will study is via *composition* of two facet inducing inequalities, the second will be via what we will call *node lifting*.

4.2. Composition of inequalities

In [616], a composition of valid inequalities known as the *s-sum* is defined. We will focus here on the case of *2-sums*. For sake of simplicity we will assume that the inequalities $fx \geq f_0$ we deal with are *simple*, that is $f_e > 0$ for all $e \in E$.

For the reader with some knowledge of the traveling salesman polytope, the $2-sum$ composition we describe here, can be used to build clique tree inequalities from comb inequalities.

We say that two edge weighted graphs $G^1 = (V^1, E^1, f^1)$ and $G^2 = (V^2, E^2, f^2)$ are *isomorphic* if there exists a one to one correspondence between their vertex sets that preserves the weights on the edges.

Definition 30 *Let $f^1x \geq f_0^1$ and $f^2x \geq f_0^2$ be two TT inequalities valid for $GTSPP(n_1) = (V^1, E^1)$ and $GTSPP(n_2) = (V^2, E^2)$, respectively. Let $e_1 = (u_1, v_1) \in E_1$ and $e_2 = (u_2, v_2) \in E_2$ be two edges such that $f_{e_1}^1 = f_{e_2}^2 = \epsilon > 0$. Then a $2 - sum$ inequality obtained by* identifying *u_1 with u_2 (called now u) and v_1 with v_2 (called now v) is the inequality $fx \geq f_0^1 + f_0^2 - 2\epsilon$ defined on $G = K_n$ with $n = n_1 + n_2 - 2$ as follows:*

1 *$V_n = (V^1 \setminus \{u_1, v_1\}) \cup (V^2 \setminus \{u_2, v_2\}) \cup \{u, v\}$*

2 *The weighted subgraph $G(V^i)$ is isomorphic to the weighted graph G^i, with u and v corresponding to u_i and v_i respectively, for $i = 1, 2$.*

3 *The coefficients of the edges with one endpoint in V^1 and the other in V^2, that we call the* crossing edges *of the $2-sum$, are computed in the following way*

   (a) *Order the crossing edges $e_1, \ldots, e_j, \ldots, e_k$*

   (b) *For $j = 1$ to k, let T^j be a minimum f-length closed walk among all closed walks of G containing e_j that uses only edges from $E_1 \cup E_2 \cup \{e_1, \ldots, e_j\}$. Let f_{e_j} be such that the f-length of T^j is $f_0^1 + f_0^2 - 2\epsilon$.*

The condition $f_{e_1}^1 = f_{e_2}^2$ is not restrictive since we can always scale one of the inequalities in order to obtain this condition.

The procedure described at point 3 of Definition 30 is called *sequential lifting* in the literature (see Padberg [640]). In general the coefficients of the crossing edges depend on the lifting sequence. If it is not the case we say that the $2 - sum$ is *stable*. Most of the known inequalities that come from $2 - sums$ are stable.

If, in the sequential lifting, the coefficients of the crossing edges are such that the minimum over all closed walks is the same as the minimum over all Hamiltonian cycles, then the $2 - sum$ inequality is called *h-liftable*.

For the sake of simplicity, when obvious, we do not mention the correspondence between the two composing graphs and the corresponding elements of the isomorphic subgraphs of the $2 - sum$.

Let $fx \geq f_0$ be an inequality supporting for $GTSPP(n)$. A vertex $v \in V$ is said to be *k-critical* for that inequality if the f-length of a minimum f-length closed walk of $K_n \setminus \{v\}$ is $f_0 - k$.

Remark 31 *In the previous definition k must satisfy $k \leq 2.min\{f_e : e \in \delta(v)\}$. We will only consider two types of vertices, the* 0-critical *and the* $\overline{k}$-critical *with $\overline{k} = 2.min\{f_e : e \in \delta(v)\}$*

We assume the $2-sum$ is performed as described in Definition 30 and the notation is the one used in that definition. The following theorem, the proof of which can be found in [618], states conditions under which two facet inducing inequalities yield, by $2-sum$, another facet inducing inequality.

Theorem 32 *Let $f^1x \geq f_0^1$ and $f^2x \geq f_0^2$ be two TT facet inducing inequalities for $GTSPP(n_1)$ and $GTSPP(n_2)$, respectively. The $2-sum$ inequality $fx \geq f_0$ is facet defining for $GSTPP(n)$ if v_1 is 2ϵ-critical for $f^1x \geq f_0^1$ and at least one of the two vertices u_2 and v_2 is 2ϵ-critical for $f^2x \geq f_0^2$.*

The next theorem is the corresponding theorem for $STSPP(n)$.

Theorem 33 *Let $f^1x \geq f_0^1$ and $f^2x \geq f_0^2$ be two TT facet inducing inequalities of $STSPP(n_1)$ and $STSPP(n_2)$, respectively. The $2-sum$ inequality $fx \geq f_0$ is facet defining for $STSPP(n)$ if it is h-liftable and:*

(a) v_1 is 2ϵ-critical for $f^1x \geq f_0^1$,

(b) $\delta(u_2)$ is f^2-connected,

and either (Case (A))

(c') u_2 is 2ϵ-critical for $f^2x \geq f_0^2$,

(d') $\delta(v_1)$ is f^1-connected,

or (Case (B))

(c") v_2 is 2ϵ-critical for $f^2x \geq f_0^2$,

(d") $\delta(u_1)$ is f^1-connected,

(e") there exists a Hamiltonian cycle $H_1 \in \mathcal{H}_{f^1}^{=}$ containing edge (u_1, v_1) and any edge $e_1 \in \Delta_{f^1}(v_1)$,

(f") there exists a Hamiltonian cycle $H_2 \in \mathcal{H}_{f^2}^{=}$ containing edge (u_2, v_2) and any edge $e_2 \in \Delta_{f^2}(v_2)$,

Proof: See [618] ■

The $2-sum$ procedure can be carried out recursively. The only difficulty is to keep track of the "criticality" of the vertices after the composition. Various technical lemmas dealing with this aspect are given in [618].

4.3. Node lifting

As mentioned earlier, when studying a given family of facet inducing inequalities for STSPP(n), it is easier to first study the simple inequalities of the class. Then the result is generalized (*lifted*) to the non simple inequalities of that family. The tools to obtain these results are the aim of this section. Remark 19 states that a facet inducing inequality in TT form is such that the the components of the subgraph on the edges with zero coefficient are cliques. This may suggest that the only *lifting* operation is the replacement of nodes by cliques. This is not the case. A comb (see next section) with no node inside the handle not contained in any tooth yields a facet inducing inequality. The same comb with a node in the handle not in any tooth also does.

Let $fx \geq f_0$ be a simple valid inequality for $GTSPP(n)$. An inequality $f^*x \geq f_0$ (same right hand side) is obtained by *clique lifting* from a simple one if each node $u \in V$ is replaced by a clique $K^u_{k_u} = (V^u, E^u)$ of $k_u \geq 1$ vertices and:

1 $f^*_e = 0$ for all $e \in E^u$ for some $u \in V$,

2 for all $e \in (V^u : V^v)$, $f^*_e = f_{uv}$, for all $u, v \in V$.

That is all the edges of the newly formed cliques get a coefficient of zero, the edges between two newly formed cliques inherit the coefficient of of the corresponding edge in the original graph.

Theorem 34 *Let $fx \geq f_0$ be a TT facet inducing inequality of $GTSPP(n)$. Any inequality obtained from $fx \geq f_0$ by clique lifting is also facet inducing for $GTSPP(n^*)$, where $n^* > n$ is the number of vertices of the resulting graph.*

Proof: The inequality is obviously valid since any closed walk on the "extended" graph induces a closed walk on K_n. It is supporting since any closed walk T of K_n can be extended to a closed walk T' on the extended graph with $f(T) = f^*(T')$. Assume $f^*x \geq f_0$ is not facet inducing. Since it is valid and supporting, there is a facet that contains the face it induces. Let $gx \geq f_0$ be a linear representation of that facet (by scaling we can always assume the right hand side to be f_0). Note that we have $\mathcal{W}^=_{f^*} \subset \mathcal{W}^=_g$. Let $e \in \gamma(V_u)$ for some $u \in V_n$ and $T' \in \mathcal{W}^=_{f^*}$, so $T' \in \mathcal{W}^=_g$. The closed walk $T' + 2\{e\}$ is also in $\mathcal{W}^=_{f^*}$ and therefore in $\mathcal{W}^=_g$, which implies that $g_e = f_e = 0$. We are therefore left with the edges which correspond to some edge of the simple inequality, and proving that $g_e = f^*_e = f_e$ for these amounts to proving that the simple inequality is facet inducing. ■

Things are not that easy for $STSPP(n)$. The clique lifting can be seen as a recursive application of an operation we will call *zero-lifting of a node*. We define now what we mean by *node lifting*.

Let $K_n = (V_n, E_n)$ and $K_{n^*} = (V_{n^*}, E_{n^*})$, $n^* \geq n$ be two complete graphs. Let $V_n = \{u_1, u_2, \ldots, u_n\}$ and $V_{n^*} = V_n \cup \{u_{n+1}, \ldots, u_{n^*}\}$. We say that the inequality $f^*x \geq f_0^*$ defined on $\mathbb{R}^{E_{n^*}}$ has been obtained by *node-lifting* of $fx \geq f_0$ defined on $\mathbb{R}^{E_n}$ if

$$f^*_{(u_i u_j)} = f_{(u_i u_j)} \text{ for all } 1 \leq i < j \leq n.$$

The special case where $f^*_{(uv)} = 0$ for some $u \in V_n$ and all $v \in \{u_{n+1}, \ldots, u_{n^*}\}$ is called *zero-lifting of node u*. Note that by triangularity one then have $f^*_{(vw)} = 0$ for all $v, w \in \{u_{n+1}, \ldots, u_{n^*}\}$. For a TT inequality obtained by zero-lifting, the following proposition is a direct consequence of Remark 19.

Proposition 35 *For any facet inducing TT inequality $f^*x \geq f_0$ obtained from $fx \geq f_0$ by zero-lifting of node u, the following holds:*

1 $f^*_{(vw)} = f_{(vu)}$ *for all* $v \in V_n - \{u\}$ *and* $w \in \{u_{n+1}, \ldots, u_{n^*}\}$,

2 $\Delta_{f^*}(w) = \Delta_f(u) \cup \bigcup\{\delta(w') \setminus \{(w', w)\} : w' \in \{u, u_{n+1}, \ldots, u_{n^*}\}, w' \neq w\}$ *for all* $w \in \{u_{n+1}, \ldots, u_{n^*}\}$, *where* $\delta(w')$ *is a taken in* E_{n^*}.

There are node-liftings which are not zero-liftings. For example the inequality of Figure 2.5 (b) is what is called in [618] a *1-node lifting* of the inequality of Figure 2.5 (a), and that of Figure 2.5 (c) a 1-node-lifting of that of Figure 2.5 (b). In that figure the coefficient of the shown edges is the one shown; for the others the coefficient is that of a shortest path, in term of coefficient of the shown edges, between their end point. For example, in Figure 2.5 (a) a non shown edge linking a node of the top cycle to one of the bottom cycle and not shown has a coefficient of 3, since a shortest path between its extremities consists of an horizontal edge and a vertical one. All right hand sides are 10. Note that these three inequalities are what we will call later on *comb inequalities*.

A *1-node lifting* of an inequality $fx \geq f_0$ defined on $\mathbb{R}^{E_n}$ is an inequality $f^*x \geq f_0$ (same right hand side) defined on $\mathbb{R}^{E_{n+1}}$ and which is not a zero-lifting. The coefficients $f^*_{vu_{n+1}}$ for $v \in V_n$, u_{n+1} being the new vertex, are not defined by this definition. They must satisfy the following necessary conditions.

Theorem 36 *Let $f^*x \geq f_0^*$ be a facet defining inequality for $GTSPP(n+1)$ which is obtained by 1-node lifting of a TT facet inducing inequality for $GTSPP(n)$; then the following conditions hold:*

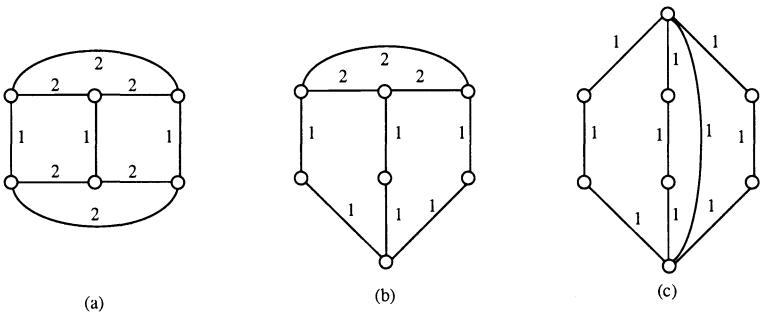

Figure 2.5. Examples of 1-node liftings

1 $f^*x \geq f_0^*$ *is tight triangular,*

2 $f_0^* = f_0$

3 for all $e \in \Delta_{f^*}(u_{n+1})$ *there exists* $e' \neq e, e' \in \Delta_{f^*}(u_{n+1})$ *and* $\Gamma \in \mathcal{H}_f^=$ *such that both* e *and* e' *belong to* Γ.

4 every connected component of the graph $(V_n, \Delta_{f^*}(u_{n+1}))$ *contains at least one odd cycle.*

Proof: By Theorem 17, the only non TT facet inducing inequalities for $GTSPP(n+1)$ are the non negativity constraints or the degree constraints, and none can be obtained by 1-node lifting, so Statement 1 follows. We can only have $f_0^* \geq f_0$. Since the inequality is TT, let $e \in \Delta_{f^*}(u_{n+1})$. There is a closed walk W of $\mathcal{W}_f^=$ that contains $e = (u, v)$, and $W' = W - \{e\} + \{(u, u_{n+1}), (v, u_{n+1})\}$ is such that $f^*(W') = f_0$, which implies $f_0^* = f_0$. For the rest see [618]. ■

Facet inducing results on 1-node lifting for $GTSPP(n)$ and $STSPP(n)$ can be found in [618]. One can also find there some results on zero-liftings for $STSPP(n)$ (we have settled here the case of $GTSPP(n)$).

We now give a 2-node lifting procedure, known as *edge cloning*, which concerns two vertices and that cannot be obtained by successive 1-node liftings.

Let $fx \geq f_0$ be a TT inequality defined on $\mathbb{R}_n^E$ and $e \in E_n$. We say that the inequality $f^*x \geq f_0^*$ defined on $\mathbb{R}_{n+2h}^E$, with $h \geq 1$, is obtained from $fx \geq f_0$ by cloning h times the edge e if it is obtained by node lifting of $fx \geq f_0$ and, assuming without loss of generality that $e = (u_{n-1}, u_n)$:

$$f_0^* = f_0 + 2hf_e,$$

$$f^*_{u_i u_{n+j}} = \begin{cases} f_{u_i u_{n-1}} & \text{for } 1 \leq i \leq n-2, 1 \leq j \leq 2h-1 \text{ and } j \text{ odd}, \\ f_{u_i u_n} & \text{for } 1 \leq i \leq n-2, 1 \leq j \leq 2h-1 \text{ and } j \text{ even}, \end{cases}$$

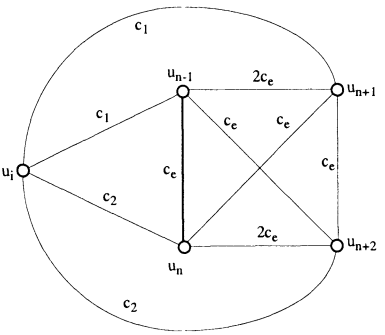

Figure 2.6. Examples of edge cloning

$$f^*_{u_{n+i}u_{n+j}} = \begin{cases} 2f_e & \text{for } -1 \leq i < j \leq 2h \text{ and } j-i \text{ even,} \\ f_e & \text{for } -1 \leq i < j \leq 2h \text{ and } j \text{ odd,} \end{cases}$$

Figure 2.6 shows the coefficients of the inequality with edge e cloned and $h = 1$. Let $fx \geq f_0$ be a TT inequality valid for $GTSPP(n)$. We say that $e = (u, v)$ is *f-clonable* if:

1 u and v are $2f_e$-critical for $fx \geq f_0$,

2 the f-length of any closed walk W of K_n is at least $f_0 + (d-2)f_e$ where $d = min(|(\delta(u) \cap T)|, |(\delta(v) \cap T)|)$.

This definition is more restrictive than that given in [618]. The first condition is not required there, but is required in all the subsequent theorems. In Figure 2.7 the coefficients of the edges are shown next to them, if not equal to 1. The edges not shown have a coefficient equal to that of a shortest path, with edges among those shown, between their extremities. The first inequality is a comb inequality, the right hand side is 10, only the three edges in bold are f-clonable. The other edges in that example are not f-clonable because the six vertices incident to the bold edges are 2-critical and the other two are 0-critical, therefore no other edge has its two extremities $2f_e$-critical. An example of violation of the second condition is given in the second example (the right hand side is 14). The second inequality is what we will call, later on, a path inequality. None of the edges is f-clonable. All the vertices except the top and bottom one are 2-critical, the other two vertices being 0-critical. In particular, vertices u and v (in black) are 2-critical. The shown closed walk W has f-value $14 = f_0$ and both extremities of e have degree 4 in W, which shows that edge e is not clonable. The case of the other edges follows by symmetry.

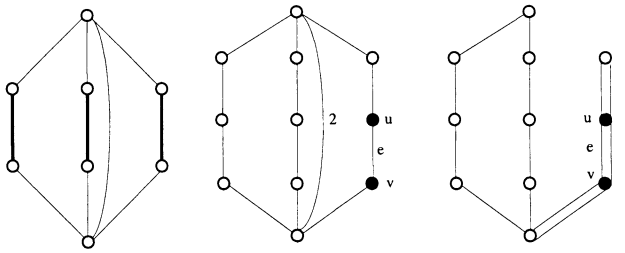

Figure 2.7. Examples of clonable and non clonable edges

Theorem 37 *Let $fx \geq f_0$ be a TT facet inducing inequality for $GTSPP(n)$ and $e = (u_{n-1}, u_n)$ be a f-clonable edge. Then the following holds:*

1 *the inequality $f^*x \geq f_0^*$, obtained from $fx \geq f_0$ by cloning $h > 1$ times edge e, is facet inducing for $GTSPP(n + 2h)$,*

2 *the edges of $(\{u_{n-1}, u_{n+1}, \ldots, u_{n+2h-1}\} : \{u_n, u_{n+2}, \ldots, u_{n+2h}\})$ are f^*-clonable,*

3 *if $e' = (z_1, z_2) \neq e$ is an edge in E_n such that z_1 and z_2 are $2f_{e'}$-critical for $fx \geq f_0$, then z_1 and z_2 are $2f_{e'}^*$-critical for $f^*x \geq f_0^*$.*

Proof: See the proofs of Lemma 4.11 and Theorem 4.12 in [618]. ■

Note that Statement 2 implies that the "new" vertices are $2f_e^*$-critical. The purpose of Statements 2 and 3 is to fix the status of the various vertices and edges in view of further edge clonings. The corresponding theorem for $STSPP(n)$ is:

Theorem 38 *Let $fx \geq f_0$ be a non-tivial TT facet inducing inequality for $STSPP(n)$ and let $e = (u_{n-1}, u_n)$ be a f-clonable edge. Then the inequality $f^*x \geq f_0^*$ obtained from $fx \geq f_0$ by cloning $h > 1$ times edge e is facet inducing for $STSPP(n + 2h)$*

Proof: See the proof of Theorem 4.13 in [618]. ■

The family of chain inequalities defined by Padberg and Hong [643] can be obtained from the comb inequalities by edge cloning.

We now turn to the study of specific classes of facet inducing inequalities of $GTSPP(n)$ and of $STSPP(n)$.

5. The Comb inequalities

Comb inequalities were first discovered, in a more restrictive form, by Chvátal [195] who derived them from the 2-matching constraints of

Edmonds [265]. They were generalized to the current form and shown to be facet inducing for $STSPP(n)$ by Grötschel and Padberg [403].

A *comb inequality* is usually defined by a set $H \subset V$, called *handle*, and an odd number $t \geq 3$ of vertex subsets $\{T_1, T_2, \ldots, T_t\}$, called *teeth*, such that:

$$H \cap T_i \neq \emptyset \quad \text{for } i = 1, ..., t \tag{14}$$

$$T_i \setminus H \neq \emptyset \quad \text{for } i = 1, ..., t \tag{15}$$

$$T_i \cap T_j = \emptyset \quad \text{for } 1 \leq i < j \leq t \tag{16}$$

Conditions (14) and (15) say that every tooth T_i intersects properly the handle H, Condition (16) states that no two teeth intersect.

The corresponding *comb inequality* is:

$$x(\delta(H)) + \sum_{j=1}^{t} x(\delta(T_j)) \geq 3t + 1 \tag{17}$$

Remark 39 *Comb inequalities are better known in the following form:*

$$x(\gamma(H)) + \sum_{i=1}^{t} x(\gamma(T_i)) \leq |H| + \sum_{i=1}^{t} (|T_i| - 1) - (t+1)/2 \tag{18}$$

This form still has a few adepts because:

- *It is a rank type inequality for the independence system associated with the monotonization of the polytope we mentioned earlier.*
- *It is a mod-2 cut, fact which is used in some separation routines (see Caprara, Fischetti and Letchford [161], Letchford [559]).*
- *It is sometimes sparser than the other form, fact which is useful when solving large linear programs.*

On the other hand it is not in TT form. We will never use this form.

Note that Inequality (17) is closed under taking complements of sets since $x(\delta(S)) = x(\delta(V \setminus S))$ for all proper subset S of V. The definition of a comb inequality is not very satisfactory since Condition (16) is not closed under taking complements. There is no real way around this problem as long as we do not deal directly with the edge sets. The most satisfactory one is to require that the given sets satisfy the conditions after eventually replacing some of them by their complements. Figure 2.8 gives two sets $\mathcal{S}$ that give the same comb inequality, the first is a set as described in (14) to (16), the other with T_3 replaced by its complement.

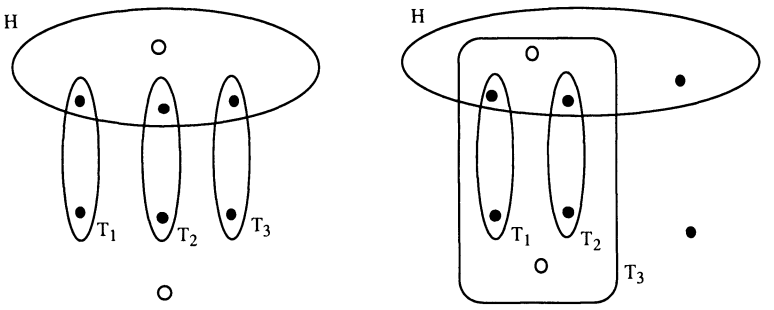

Figure 2.8. Example of a comb ($t = 3$)

Figure 2.8 enables us to define our convention for figures that will hold throughout this chapter. Sets with a black point have to be nonempty, those with a white filled point may or may not have nodes in them, those with no points must be empty. A point does not represent a unique node but a set of nodes in that position.

We will give three proofs of validity of comb inequalities, an algebraic (classical) one and two others that will give more insight into what these inequalities say in term of closed walks. We have so far used the term *tight* in various ways. We now use it for sets and will use it frequently. A subset $S \subset V$ is said to be *tight* for a Hamiltonian cycle or a closed walk Γ if $|\Gamma \cap \delta(S)| = 2$. In other terms, if x^Γ is the incidence vector of Γ, the corresponding subtour elimination inequality is tight, i.e. $x^\Gamma(\delta(S)) = 2$.

Theorem 40 *Comb inequalities are valid for GTSPP(n) (and therefore for STSPP(n)).*

Proof: **(1)** It is easy to check that we have:

$$x(\delta(H)) + \sum_{j=1}^{t} x(\delta(T_j)) \geq \quad (19)$$

$$1/2\left[\sum_{j=1}^{t} x(\delta(H \cap T_j)) + \sum_{j=1}^{t} x(\delta(T_j \setminus H)) + \sum_{j=1}^{t} x(\delta(T_j))\right] \geq 3t$$

Now the left part of (19), $x(\delta(H)) + \sum_{j=1}^{t} x(\delta(T_j))$, is an even integer as sum of even integers, when x represents a closed walk. Since $3t$ is odd, $3t$ can be replaced by $3t + 1$. ■

This easy proof does not really explain why any Hamiltonian cycle or any closed walk, intersects these $t+1$ coboundaries in at least $3t+1$ edges and not in $2t+2$ obtained by adding up the $t+1$ corresponding subtour elimination inequalities. The following proof, taken from Naddef and Pochet [615], will give more insights into the validity of comb inequalities.

Proof: (2) A non-algebraic and inductive proof of validity of comb inequalities. The comb inequality is assumed to be described by sets that satisfy Conditions (14) to (16). Assume first that t=3. Take a Hamiltonian cycle Γ, if $|\Gamma \cap \delta(T_j)| = 2$ for j=1, 2 and 3, then $\Gamma \cap (T_i \setminus H : T_i \cap H) \neq \emptyset$, so $|\Gamma \cap \delta(H)| \geq 3$. Since $|\Gamma \cap \delta(H)|$ must be even we have $|\Gamma \cap \delta(H)| \geq 4$ and Inequality (17) is satisfied by all x representing such a cycle. If $|\Gamma \cap \delta(T_{j^\star})| \neq 2$, for some $j^\star$, $|\Gamma \cap \delta(T_{j^\star})| \geq 4$, the 3 other sets having at least 2 edges of their coboundaries in Γ, again Inequality (17) is satisfied by all x representing such a cycle. Proving validity for the general case can be done by induction on the number of teeth $t \geq 5$. Assume validity for combs with $t-2$ teeth. Take a Hamiltonian cycle Γ. If $|\Gamma \cap \delta(T_j)| = 2$ for $j = 1, \ldots, t$, then $|\Gamma \cap \delta(H)| \geq t$ and by parity one must have $|\Gamma \cap \delta(H)| \geq t+1$ and the result follows for those cycles. Else assume $|\Gamma \cap \delta(T_{j^\star})| \geq 4$, for some $j^\star$, let us assume for simplicity $j^\star = t$. By induction hypothesis we have that: $|\Gamma \cap \delta(H)| + \sum_{j=1}^{t-2} |\Gamma \cap \delta(T_j)| \geq 3t - 6 + 1$, adding $|\Gamma \cap \delta(T_t)| \geq 4$ and $|\Gamma \cap \delta(T_{t-1})| \geq 2$ we prove again that the inequality is valid. ■

This second proof shows why the teeth and the handle play a different role. The teeth, in some sense, "force" a certain number of edges to be present in the coboundary of the handle.

The last proof is given because it may be adapted to prove validity of most of the inequalities of the coming sections. It is similar to a proof that can also be found in Applegate et al. [29].

Proof: (3) A non-algebraic and non-inductive proof of validity of comb inequalities. Let Γ be any Hamiltonian cycle, and x^Γ its associated characteristic vector. Inequality (19) is satisfied by x^Γ if all teeth are tight for Γ using exactly the same argument as in the previous proof. It is also trivially satisfied whenever $x^\Gamma(\delta(H)) \geq t + 1$. In the other cases, i.e. $x^\Gamma(\delta(H)) < t + 1$, define η by $x^\Gamma(\delta(H)) = t + 1 - 2\eta$. Note that $\eta > 0$ is integer because t is odd, and $x^\Gamma(\delta(H))$ even. As $(t+1)/2$ represents the minimum number of paths traversing H when all teeth are tight, η represents the reduction in number of paths traversing H with respect to the tight teeth case. We prove basically, and the comb inequality tells us essentially, that the number of non tight teeth has to be at least as large as η.

First, we consider each tooth T_j separately, and define $\theta_j = 1$ if $x^\Gamma(T_j \cap H : T_j \setminus H) = 0$, $\theta_j = 0$ otherwise. It is readily seen that $x^\Gamma(\delta(T_j))/2 \geq 1 + \theta_j$ because $x^\Gamma(\delta(T_j))/2$ is the number of paths traversing T_j in the tour Γ, and the tooth T_j is traversed by at least two paths (i.e. is not tight) when $\theta_j = 1$. Thus, $x^\Gamma(\delta(T_j)) - 2 \geq 2\theta_j$.

Next, we consider the handle for which it is easy to check that $x^\Gamma(\delta(H)) \geq \sum_{j=1}^{t} x^\Gamma(T_j \cap H : T_j \setminus H) \geq \sum_{j=1}^{t}(1-\theta_j) = t - \sum_{j=1}^{t}\theta_j$. Together with $x^\Gamma(\delta(H)) = t+1-2\eta$, this gives $\sum_{j=1}^{t}\theta_j \geq 2\eta - 1 \geq \eta$ where the last inequality holds because η is a positive integer.

Hence, we have shown that the reduction η in the number of paths traversing H is at most $\sum_{j=1}^{t}\theta_j$, which is at most, by definition of θ_j, the number of non-tight teeth. This suffices to prove validity because $x^\Gamma(\delta(H)) = t+1-2\eta \geq t+1-2\sum_{j=1}^{t}\theta_j \geq t+1-\sum_{j=1}^{t}[x^\Gamma(\delta(T_j))-2]$. ■

Theorem 41 *(Grötschel and Padberg [403], [404]) The comb inequalities are facet inducing for $STSPP(n)$ and therefore for $GTSPP(n)$ $(n \geq 6)$.*

Proof: The original proof is long and technical. It can be rewritten by proving that the simple comb inequalities are facet inducing for $GTSPP(n)$ (very easy), then for $STSPP(n)$ by using Lemma 29, and finally using the node lifting theorems of [618] to obtain the result for general comb inequalities. ■

Comb inequalities together with subtour elimination inequalities and non negativity inequalities, completely describe $STSPP(6)$, but not $STSPP(7)$ (see end of next section). Comb inequalities, as we will see, are central in solving large TSP instances to optimality.

We now turn to more classes of inequalities. For most of them, comb inequalities can be seen as a subfamily. For the others, the comb inequalities can be seen as building blocks.

6. The Star and Path inequalities

Star inequalities (Fleischman [311]) are defined by two sets of subsets of V, $\mathcal{K} = \{H_1, \ldots, H_h\}$, called *handles* and $\mathcal{T} = \{T_1, \ldots, T_t\}$, called *teeth*, with $t \geq 3$ and odd, together with non-zero integers $\alpha_1, \ldots, \alpha_h$ associated with the handles, and non-zero integers $\beta_1, \ldots, \beta_t$ associated with the teeth, such that:

$$H_1 \subset H_2 \subset \cdots \subset H_i \subset \cdots \subset H_h \tag{20}$$

$$T_i \cap T_j = \emptyset \qquad \text{for } 1 \leq i < j \leq t \tag{21}$$

$$H_1 \cap T_j \neq \emptyset \qquad \text{for } j = 1, \ldots, t \tag{22}$$

$$T_j \setminus H_h \neq \emptyset \qquad \text{for } j = 1, \ldots, t \tag{23}$$

$$(H_{i+1} \setminus H_i) \setminus \bigcup_{j=1}^{t} T_j = \emptyset \qquad \forall i, 1 \leq i \leq h-1 \tag{24}$$

And a condition that relates the integers β_j to the integers α_i, which we call the *Interval Property*.

To define that property we will need a few definitions but we give first the corresponding Star inequality:

$$\sum_{i=1}^{h} \alpha_i x(\delta(H_i)) + \sum_{j=1}^{t} \beta_j x(\delta(T_j)) \geq (t+1) \sum_{i=1}^{h} \alpha_i + 2 \sum_{j=1}^{t} \beta_j \qquad (25)$$

Given a tooth T_j, a maximal index set of (successive) handles which have the same intersection with T_j is called *interval* relatively to T_j . We call $\sum_{i=\ell}^{\ell+r} \alpha_i$ the *weight* of the interval $I = \{\ell, \ell+1, ..., \ell+r\}$.

Definition 42 ***The Interval Property****: for each tooth T_j, we have β_j at least equal to the maximum weight of an interval relative to T_j*

Figure 2.9 shows a tooth and the traces of the different handles on it. In the tooth, where a node is shown, there is at least one node, where none is shown there are no nodes. The handles, as suggested by the drawing, are numbered from top to bottom. There are 4 intervals, $\{1\}, \{2,3\}, \{4,5\}, \{6,7\}$, of weight 3, 4, 5 and 4, and therefore the β coefficient of that tooth must be at least 5.

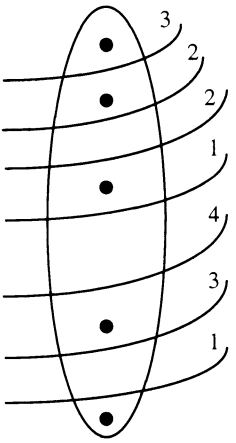

Figure 2.9. Example of intervals

Path inequalities (see [219]) are the special case where all the intervals relative to a same tooth have the same weight and the coefficient of that tooth is exactly that value. Figure 2.10 gives an example of a Path inequality. The coefficients associated with the handles and teeth are beside each set.

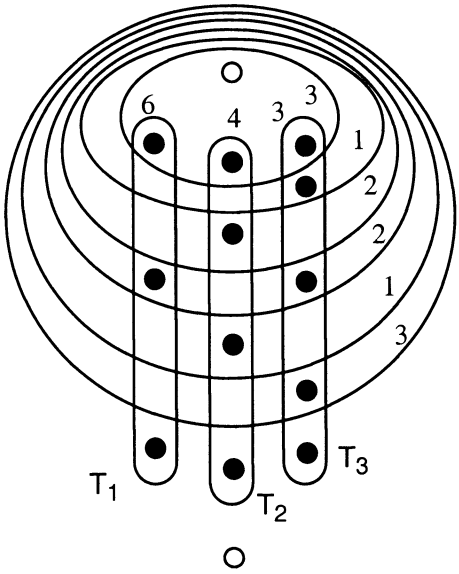

Figure 2.10. Example of a path ($h = 6$, $t = 3$)

In [219] a distinction is made between the case in which there is at least one vertex in the position of each of the two white filled points (path inequalities), only in one of these positions (wheelbarrow inequalities) or in none of these two positions (bicycle inequalities). We will here represent all three case under the same name: *path inequality.* Comb inequalities are exactly the path inequalities with a single handle.

The rationale behind the Interval Property is far from being obvious. We will try to explain it.

The RHS of Inequality (25) can easily be understood by the following argument. Consider Hamiltonian cycles for which all teeth are tight. Each handle must then have at least $t + 1$ edges of those cycles in its coboundary because t is odd (see comb inequalities), which leads to the first part of the RHS. The second part is due to the coboundaries of the teeth. This shows that for all such tours the inequality is valid. As one may suspect, the Interval Property is designed to make it valid for all the other tours. This property can be understood by looking at the tours Γ that are tight on all teeth except T_j and $|\Gamma \cap \delta(T_{j^\star})| = 4$. Let $\{r, \cdots, s-1\}$ be an interval of $T_{j^\star}$. Assume Γ uses edges of the coboundaries of the handles H_r to H_{s-1} only inside the teeth T_i for $i \neq j^\star$. In other words, $x(T_{j^\star} \cap H_i : T_{j^\star} \setminus H_i) = 0$ for all $i \in \{r, \cdots, s-1\}$ (see Figure 2.11 where $j^\star = 2$ from Figure 2.10). Then the coboundaries of the handles H_r to H_{s-1} may contain only $t-1$ edges, the coboundaries

of the other handles only $t+1$. Therefore a loss to the LHS, relatively to the former case, of $2\sum_{i=r}^{s-1}\alpha_i$ is compensated by an increase of $2\beta_{j^\star}$ due to the tooth $T_{j^\star}$ having now four edges in its coboundary. Therefore, to have validity of the inequality one must have $\beta_{j^\star} \geq \sum_{i=r}^{s-1}\alpha_i$. This argument is valid for all intervals of $T_{j^\star}$ and therefore $\beta_{j^\star}$ must be greater or equal to the largest weight of an interval.

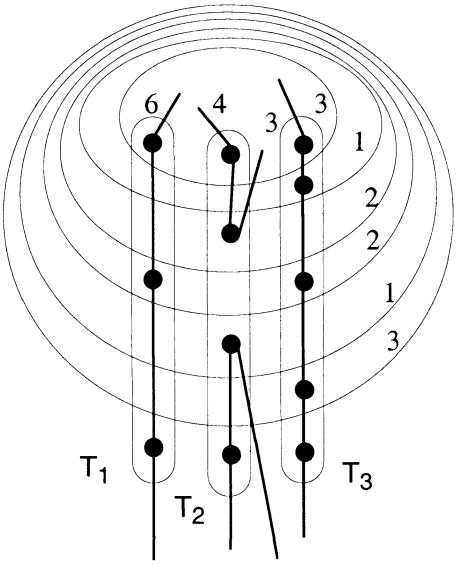

Figure 2.11. Example of minimal trace with T_2 not tight ($r = 3$, $s - 1 = 4$)

Of course, all this does not constitute a proof of validity. One can adapt the third proof of validity of combs to obtain one for the stars (see [615]). See [311] for a proof of:

Theorem 43 *Star inequalities are valid for $GTSP(n)$ and therefore for $STSPP(n)$.*

Most star inequalities are not facet inducing even for $GTSP(n)$, but we have:

Theorem 44 *Path inequalities are facet inducing for $GTSP(n)$.*

Proof: See [219] ■

In Naddef and Rinaldi [619], and unpublished work of Queyranne and Wang the corresponding theorem for $STSPP(n)$ can be found.

In the theoretical study of Goemans [383] on the comparison of valid inequalities for the TSP, the path inequalities turn out to be, at least in theory, the most powerful known inequalities in term of potential to

reduce the gap between the Held and Karp bound and the optimal value. We will see that they do turn out to be also useful in practice.

We end this section with an example of a facet inducing star inequality for $STSPP(n)$ which is not a path inequality. It is shown in Figure 2.12.

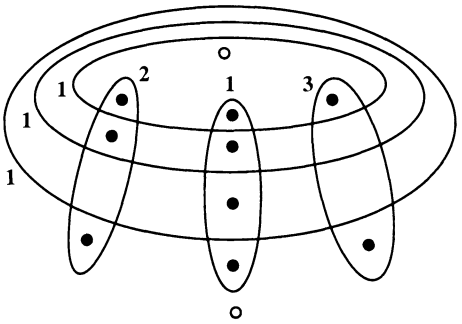

Figure 2.12. A facet inducing star (non-path) inequality; RHS=24

Research Question 45 *Find a necessary and sufficient condition for a star inequality to be facet inducing for $GTSPP(n)$.*

Chances are that this necessary and sufficient condition will also hold for $STSPP(n)$. It is easy to find necessary conditions. One is that the interval property must be satisfied with equality, else the inequality is the sum of another star inequality and of a subtour elimination inequality. Another necessary condition is that one cannot decompose the coefficients α_i and β_j in such a way that the star inequality is the sum of two other star inequalities.

To give an idea of the difficulty of the question, the first star inequality of Figure 2.13 is not facet inducing for $GTSPP(n)$, $n \geq 8$, while the second is for $n \geq 10$. These two inequalities look pretty much like that of Figure 2.12. The RHS are 26 and 22, respectively.

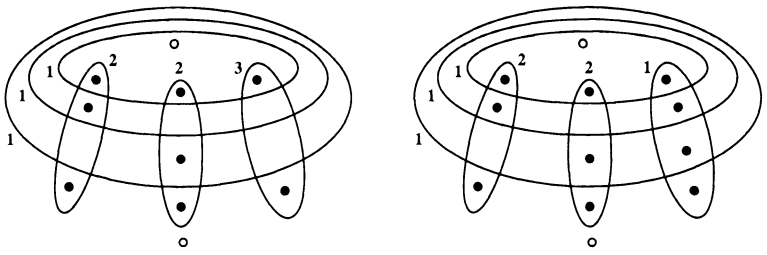

Figure 2.13. Examples of non facet and facet inducing star inequalities

For $n = 7$ path inequalities together with comb inequalities, subtour elimination inequalities and non-negativity constraints define entirely $STSPP(7)$. For $n = 8$, to obtain such a description, one must add the crown inequalities ([617]) and four classes known as NEW1, NEW2, NEW3 and NEW4 that can be found in [473].

7. The Clique Tree and Bipartition inequalities

Clique tree inequalities were defined by Grötschel and Pulleyblank [406]. A *Clique tree* is defined by two sets of subsets of V, $\mathcal{K} = \{H_1, \ldots, H_h\}$ and $\mathcal{T} = \{T_1, \ldots, T_t\}$, such that:

$$H_i \cap H_j = \emptyset \qquad \text{for } 1 \leq i < j \leq h \qquad (26)$$

$$T_i \cap T_j = \emptyset \qquad \text{for } 1 \leq i < j \leq t \qquad (27)$$

$$T_j \setminus \bigcup_{i=1}^{h} H_i \neq \emptyset \qquad \text{for } 1 \leq j \leq t \qquad (28)$$

$$t_j = |\{i : T_j \cap H_i \neq \emptyset\}| \geq 1 \qquad \text{for } 1 \leq j \leq t \qquad (29)$$

$$h_i = |\{j : H_i \cap T_j \neq \emptyset\}| \geq 3 \text{ and odd} \qquad \text{for } 1 \leq i \leq h \qquad (30)$$

$$\text{If } T_j \cap H_i \neq \emptyset, \text{it is a disconnecting set of the hypergraph } (V, \mathcal{K} \cup \mathcal{T}). \qquad (31)$$

In other words, handles are pairwise disjoint, and so are the teeth. No tooth is contained in a handle, every tooth has at least a vertex not in any handle and every handle intersects an odd number (≥ 3) of teeth. The *tree-like* structure is given by Condition (31). Further, in this section, we will drop this last condition together with Condition (28) to obtain a larger family of inequalities, the *bipartition inequalities* of Boyd and Cunningham [135]. When studying these inequalities, we will see the role of Condition (28) in the definition of a clique tree.

Figure 2.14 gives an example of a clique tree. Handles are in bold lines. If there are no vertices in position "Z", then the clique tree spans the whole graph.

The corresponding *clique tree inequality* is:

$$\sum_{i=1}^{h} x(\delta(H_i)) + \sum_{j=1}^{t} x(\delta(T_j)) \geq \sum_{i=1}^{h} (h_i + 1) + 2t = 2h + 3t - 1 \qquad (32)$$

in which h_i is defined by Condition (30).

As for all the inequalities that we will propose in this chapter, the RHS can easily be figured out by looking at tours for which all teeth are tight. For these tours, the handles must have at least $h_i + 1$ edges in their coboundaries, and this leads naturally to the RHS. To prove validity, one must prove that every time we augment the number of

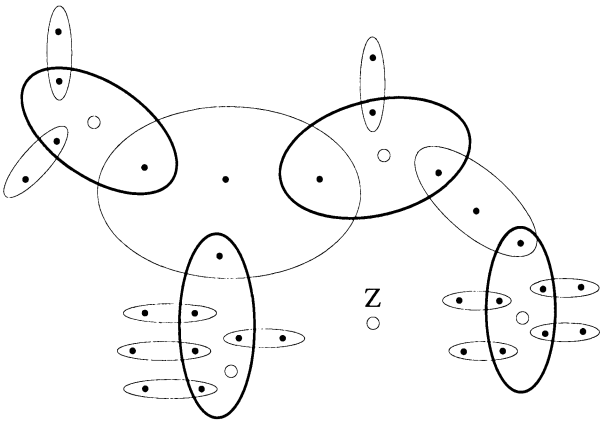

Figure 2.14. Example of a Clique Tree

edges in the coboundary of some teeth, we do not "save" more on the total number of edges in the coboundary of the handles. This is done in proofs of Applegate et al. [29] and in Naddef and Pochet [615]. An algebraic proof of validity is given in [406]. Moreover one can find there the following theorem:

Theorem 46 *(Grötschel-Pulleyblank [406]) Clique tree inequalities are facet inducing for STSPP(n).*

A clique tree with a single handle is a comb. Some authors admit $\mathcal{K} = \emptyset$ (which, by the way, does not satisfy Condition (29)), if so, subtour elimination inequalities are also clique tree inequalities. In [616], Naddef and Rinaldi show that the clique trees, which have at least a vertex in position "Z" of Figure 2.14, can be obtained by repeated 2-sums starting with two combs.

We now turn to the *bipartition inequalities* of Boyd and Cunningham [135]. A *bipartition* is defined just like a clique tree but without Conditions (28) and (31). There are coefficients $\beta_1, \ldots, \beta_j, \ldots, \beta_t$ associated with the teeth. Let D be the set of indices of teeth that contain no vertex outside any handle, i.e.

$$T_j \setminus \bigcup_{i=1}^{n} H_i = \emptyset \text{ if and only if } j \in D \tag{33}$$

A tooth in the set $\{T_j : j \in D\}$ is called *degenerate* by Boyd and Cunningham. We will define, later on, a more restrictive notion of degeneracy which will not consider as degenerate those teeth with index in D which intersect only two handles. The intuitive explanation of the co-

efficients, further down, will explain why we will not qualify these teeth as degenerate.

Let t_j be as defined in Condition (29), that is the number of handles that tooth T_j intersects, then the coefficients β_j are defined by:

$$\beta_j = 1 \quad \text{if } j \notin D \tag{34}$$

$$\beta_j = \frac{t_j}{t_j - 1} \quad \text{if } j \in D \tag{35}$$

The corresponding *bipartition inequality* is:

$$\sum_{i=1}^{h} x(\delta(H_i)) + \sum_{j=1}^{t} \beta_j x(\delta(T_j)) \geq \sum_{i=1}^{h} (h_i + 1) + 2 \sum_{j=1}^{t} \beta_j \tag{36}$$

In Figure 2.15, in which handles are in bold lines, the two teeth T^* and T are degenerate. The first one intersects three handles, therefore its coefficient is 3/2, the second only two and therefore its coefficient is 2.

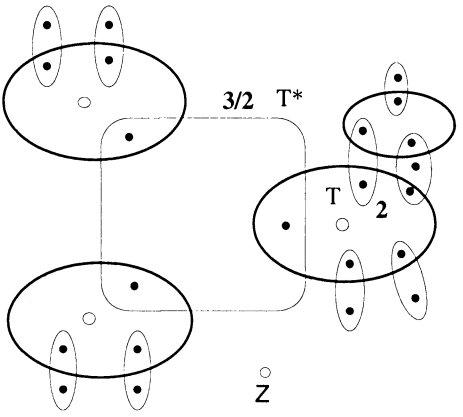

Figure 2.15. Example of a Bipartition

The right hand side is again understood by looking at the tours for which all teeth are tight. An example of the trace of such a tour on the various coboundaries, involved in the inequality, is given in the first drawing of Figure 2.16, which minimizes the total number of edges in these coboundaries. In such a tour each handle must have at least $h_i + 1$ edges, where h_i is the number of teeth that handle H_i intersects.

Assume that we look at tours which intersect tooth T in four edges. The second drawing of Figure 2.16 shows the trace of such a tour. Note that the increase of two edges in the coboundary of T is compensated for by a decrease of two edges in each of the coboundaries of the two only

handles it intersects. Enabling tours to use more edges in the coboundary of T does not yield any further gain in the minimum number of edges of the coboundaries of the handles. This explains the coefficient of 2, in order to maintain validity of the inequality.

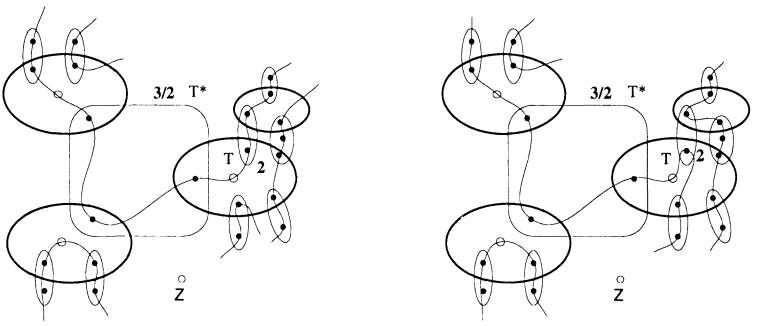

Figure 2.16. Traces of tight tours with 2 and 4 edges in $\delta(T)$

Now we turn to tooth T^*. The first drawing of Figure 2.17 shows again a minimal trace on the handles of a tour tight for all teeth. In the second, the trace corresponds to tours that intersect the coboundary of tooth T^* in four edges. We also gain two edges on the coboundaries of the handles. If we go one step further, and enable six edges in the coboundary of tooth T^*, that is four more than the minimum, this time we gain *four* edges in the coboundaries of the handles from the previous tour and *six* from the first one. If the tours, the traces of which are shown in the first and third drawing of Figure 2.17, are to be tight for the inequality, then the loss of 4 edges on the coboundary of T^* cannot be more than compensated by the gain of 6 edges on the coboundaries of the handles, therefore $\beta_{T^*} \geq 6/4 = 3/2$. Since enabling yet more edges in the coboundary of T^* does not decrease further the minimum number of edges in the coboundaries of the handles, if $\beta_{T^*} > 3/2$, then the only tight tours would be those which are tight on T^*, and therefore the inequality would be contained in the facet $x(\delta(T^*)) \geq 2$. This explains the coefficient $t_j/(t_j - 1)$ of some teeth.

In Section 9, the technique that we just used in order to define the coefficients of T and T^* will be developed into a general procedure. In some sense, the coefficient of teeth should *normally* be defined by the case of four edges in their coboundary, so we will call degenerate only those teeth for which the coefficient is defined by tours having six or more edges in their coboundaries. Therefore tooth T of our example will no longer be considered as degenerate. We make this more precise in Section 9.

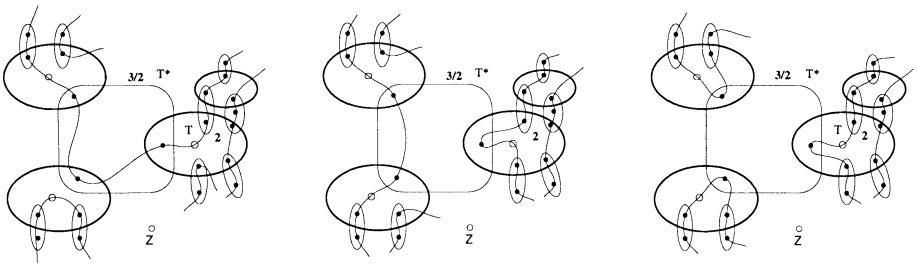

Figure 2.17. Traces of tight tours with 2, 4 and 6 edges in $\delta(T^*)$

Of course, since all we use is that a cycle intersects a coboundary in an even number of edges, so do closed walks, in all the preceding material, tours can be replaced by closed walks. In no case have we given a proof of validity of the bipartition inequality. This is done by:

Theorem 47 *(Boyd-Cunningham [135]) Bipartition inequalities are valid for STSPP(n).*

In [232] W. Cunningham and Y. Wang give two necessary conditions for such an inequality to be facet defining. The first is that there be no subset of degenerate teeth which is disconnecting for the underlying hypergraph (as a consequence, the bipartition of our example is not facet inducing). The second is that for each edge there exists a tight tour that contains it, which as already mentioned is necessary for any non-tivial inequality to be facet inducing. The next section gives an example of a bipartition which does not yield a facet inducing inequality, but for which there is a way around, via a correcting term, to get a facet inducing inequality. Cunningham and Wang conjecture that together these two conditions are sufficient.

8. The Ladder inequalities

A *ladder inequality* is defined by two handles H_1 and H_2 an *even* number $t \geq 4$ of teeth $\{T_1, \ldots, T_j, \ldots, T_t\}$, with integer coefficients $\beta_1, \ldots, \beta_j, \ldots, \beta_t$ associated with them, such that (see Figure 2.18 in

which the coefficients β are written next to the corresponding teeth):

$$H_1 \cap H_2 = \emptyset \tag{37}$$

$$T_j \cap T_k = \emptyset \qquad 1 \leq j < k \leq t \tag{38}$$

$$H_1 \cap T_1 \neq \emptyset, H_2 \cap T_1 = \emptyset \tag{39}$$

$$H_2 \cap T_2 \neq \emptyset, H_1 \cap T_2 = \emptyset \tag{40}$$

$$T_1 \setminus H_1 \neq \emptyset, T_2 \setminus H_2 \neq \emptyset \tag{41}$$

$$T_j \cap H_i \neq \emptyset \qquad \text{for } i = 1, 2 \text{ and } j \geq 3 \tag{42}$$

$$\text{if } T_j \setminus (H_1 \cup H_2) \neq \emptyset, \text{ then } \beta_j = 1, \quad \text{else } \beta_j = 2; \text{ for } j \geq 3 \tag{43}$$

$$\beta_1 = \beta_2 = 1 \tag{44}$$

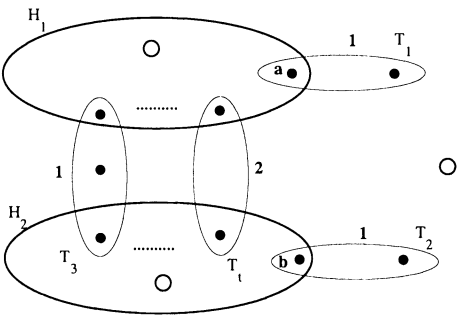

Figure 2.18. Example of a ladder

Teeth T_1 and T_2 play a special role. The first one only intersects H_1 and the second only H_2, all other teeth intersect both handles. The ladder inequality is:

$$\sum_{i=1}^{2} x(\delta(H_i)) + \sum_{j=1}^{t} \beta_j x(\delta(T_j)) - 2x(H_1 \cap T_1 : H_2 \cap T_2) \geq 2t + 2\sum_{j=1}^{t} \beta_j \tag{45}$$

The coefficients on the teeth can be explained just as in the case of the bipartition inequalities. Note the special correcting (or lifting) term $-2x(H_1 \cap T_1 : H_2 \cap T_2)$ on the left hand side of Inequality (45) without which this would only be a particular case of bipartition inequalities.

Again we try to give the reader some insight into the validity of this inequality. Observe first that no tour that is tight for the corresponding bipartition (not ladder) inequality contains an edge of $(H_1 \cap T_1 : H_2 \cap T_2)$ (we leave it to the reader to convince himself/herself of this fact or consult [615]), and therefore the corresponding face is contained in the intersection of the facets $x_e \geq 0$ for $e \in (H_1 \cap T_1 : H_2 \cap T_2)$. Figure 2.19

gives the trace of a tight tour for the ladder inequality, that contains such an edge and with all teeth tight. Extension of this tour to the case $t \geq 6$ is straightforward.

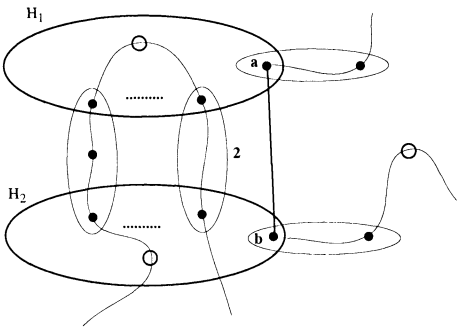

Figure 2.19. Trace of a tight tour containing $e \in (H_1 \cap T_1 : H_2 \cap T_2)$

Theorem 48 *(Boyd, Cunningham, Queyranne and Wang [136]) The ladder inequalities are facet defining for STSPP(n).*

9. A general approach to some TSP valid inequalities

Let $\mathcal{S} = \{S_1, ..., S_i, ..., S_p\}$ be a set of distinct (possibly overlapping) subsets of V. Let $\alpha_1, ..., \alpha_i, ..., \alpha_p$ be strictly positive integers. An inequality is said to be in *closed-set form* if it can be written as:

$$\sum_{i=1}^{p} \alpha_i x(\delta(S_i)) \geq r(\mathcal{S}) \tag{46}$$

where $r(\mathcal{S})$ is the minimum of the LHS on all tours.

Path, star, bipartition inequalities are inequalities in closed-set form, and therefore so are the comb and clique tree inequalities. Inequalities containing path, star and clique trees, and named *binested inequalities* in [613], are also inequalities in closed-set form. Ladder inequalities are not in closed-set form because of the correcting term on the LHS.

In most of the known closed-set form inequalities for $STSPP(n)$, the set $\mathcal{S}$ of subsets of V is partitioned into two sets $\mathcal{K}$ and $\mathcal{T}$ where $\mathcal{K} = \{H_1, \ldots, H_h\}$ is the set of so called *handles*, $\mathcal{T} = \{T_1, \ldots, T_t\}$ is the set of so called *teeth.*

We also have non-zero integers $\alpha_1, \ldots, \alpha_h$, associated with the handles, and (not necessarily integer) rationals $\beta_1, \ldots, \beta_t$, associated with the teeth.

These inequalities in closed-set form read:

$$\sum_{i=1}^{h} \alpha_i x(\delta(H_i)) + \sum_{j=1}^{t} \beta_j x(\delta(T_j)) \geq A + 2\sum_{j=1}^{t} \beta_j \tag{47}$$

or equivalently

$$\sum_{i=1}^{h} \alpha_i x(\delta(H_i) \ \geq \ A \ - \ \sum_{j=1}^{t} \beta_j \ (x(\delta(T_j)) - 2). \tag{48}$$

Expression (48) suggests the following interpretation, which can be used in order to define a general procedure for computing the coefficients A and β_j, $j = 1, \cdots, t$, of the valid inequality.

When all teeth are tight for a given Hamiltonian cycle Γ that is $x^{\Gamma}(\delta(T_j)) = 2$ for all $j = 1, \cdots, t$, Inequality (48) then reduces to $\sum_{i=1}^{h} \alpha_i x(\delta(H_i)) \geq A$. Hence, for validity, the largest possible value of A is the minimum value of the left hand side of (48) over all Hamiltonian cycles Γ with all teeth tight. Formally,

$$A \ = \ \min_{\Gamma} \{\sum_{i=1}^{h} \alpha_i |\Gamma \cap H_i| \ : \ |\Gamma \cap \delta(T_j)| = 2 \text{ for all } j = 1, \cdots, t\} \tag{49}$$

and does not depend on the coefficients β_j.

Now we describe a general procedure to compute the coefficients β_j in the valid Inequality (48) when the coefficients of the handles are assumed to be fixed. First we order the teeth. We assume without loss of generality that the teeth have been renumbered in such a way that their order is $T_1, \cdots, T_j, \cdots, T_t$. Next, we compute the coefficients β_j from $j = 1$ to $j = t$ using a sequential lifting procedure. The lifted variables are the non-negative integer slack variables s_j associated to the subtour elimination inequalities defined on the teeth, namely $s_j = x(\delta(T_j)) - 2$ for $j = 1, \cdots, t$. Initially, to compute the coefficient A in Expression (49), all slacks s_j are projected to zero and we obtain the inequality $\sum_{i=1}^{h} \alpha_i x(\delta(H_i) \geq A$ which is valid for the subspace where $s_j = 0$ for $j = 1, \cdots, t$, that is when all teeth are tight. Then, sequentially, from $j = 1$ to $j = t$, all slacks s_j are lifted.

For $k \in \{1, \cdots, t\}$, assuming coefficients $\beta_1, \cdots, \beta_{k-1}$ have been already computed, the lifting coefficient β_k is taken as the minimum value β for which the inequality

$$\sum_{i=1}^{h} \alpha_i x^{\Gamma}(\delta(H_i)) + \sum_{j=1}^{k-1} \beta_j [x^{\Gamma}(\delta(T_j)) - 2] + \beta [x^{\Gamma}(\delta(T_k)) - 2] \geq A \tag{50}$$

is satisfied by all Hamiltonian cycles Γ with $s_j = x^\Gamma(\delta(T_j)) - 2 = 0$, for all $j > k$.

We let $\mathcal{B}_\ell^k$ denote the set of Hamiltonian cycles for which all the teeth $T_{k+1}, \ldots, T_t$ are tight and intersect the coboundary of T_k in 2ℓ edges. Then the value of β_k is defined by

$$\beta_k = \max_{\ell > 1} \frac{b_k^1 - b_k^\ell}{2\ell - 2} \tag{51}$$

where

$$b_k^l = \min_{\Gamma \in \mathcal{B}_l^k} \left[\sum_{i=1}^{h} \alpha_i x^\Gamma(\delta(H_i)) + \sum_{j<k} \beta_j [x^\Gamma(\delta(T_j)) - 2] \right] \tag{52}$$

We call *degenerate* those teeth T_k for which the coefficient β_k cannot be obtained for $\ell = 2$ in Equation (51).

Note that by construction of β_k, we must have $b_k^1 = A$ for all $k \in \{1, \cdots, t\}$. Also note that the β_k are not necessarily integer. The construction is summarized in the following theorem.

Theorem 49 *Let $\mathcal{K} = \{H_1, \ldots, H_h\}$, let $\mathcal{T} = \{T_1, \ldots, T_t\}$, be an ordered set of teeth, and let $\alpha_1, \ldots, \alpha_h$ be nonzero integers associated with the handles. If A is defined by Expression (49), and, for $k \in \{1, \cdots, t\}$, if β_k is the coefficient of tooth T_k determined by Expressions (52) and (51), then the following inequality is valid:*

$$\sum_{i=1}^{h} \alpha_i x(\delta(H_i)) \geq A - \sum_{j=1}^{t} \beta_j \big[x(\delta(T_j)) - 2 \big] \tag{53}$$

Proof: Trivial by the way we constructed the coefficients of the teeth. ■

Remark 50 *The above procedure provides a way of defining valid inequalities for the TSP. The difficulty in using Theorem 49 to prove validity of inequalities is the exponential growth in the number of cases to study when the number of teeth increases.*

Remark 51 *In all the inequalities seen so far in this chapter, the coefficients of the teeth are order independent, that is they can be computed as if they are the first in the sequential lifting. This is not always the case. We give now an example.*

Figure 2.20, which has been provided by M. Queyranne, illustrates this. Here, the lifting coefficients are defined as in Theorem 49, but we

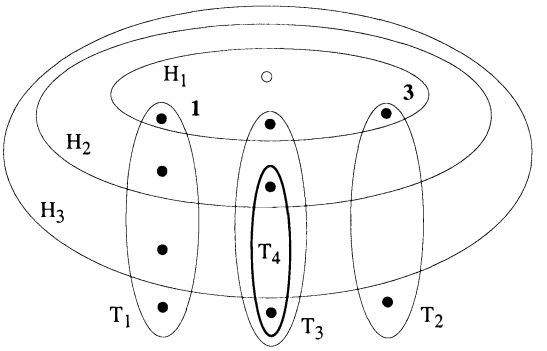

Figure 2.20. Example of a facet defining inequality

will see that their values are sequence dependent, and in fact only one order will lead to an inequality. Handles have all coefficients of 1.

The support in Figure 2.20 would give an inequality of the form

$$\sum_{i=1}^{3} x(\delta(H_i)) \geq 12 \; - \; \sum_{j=1}^{4} \beta_j \left[x(\delta(T_j)) - 2 \right]. \tag{54}$$

If you authorize a Hamiltonian cycle to intersect the coboundary of T_4 in 4 or more edges restricting all the other teeth to be tight, we cannot compensate by "saving" anything on the coboundaries of the handles. Therefore, this procedure will always produce an inequality which is just the addition of the inequality obtained without tooth T_4 and the subtour elimination inequality associated to that tooth. In other words, lifting T_4 first gives $\beta_4 = 0$.

The coefficients of T_1 and T_2 can be defined in the manner used so far. This gives $\beta_1 = 1$ and $\beta_2 = 3$. Let us authorize a Hamiltonian cycle to intersect the coboundary of T_3 in 4 edges, imposing T_4 to be tight. We can only save 2 edges on the coboundaries of the handles (whether or not we impose T_1 and T_2 to be tight); see Figure 2.21(a). These 2 edges we save are 2 of the 4 that crossed the border of H_1 when we imposed all teeth to be tight. This yields a coefficient of $\beta_3 = 1$ for T_3, instead of a coefficient of 2 in the star inequality made up of the same handles and teeth T_1, T_2 and T_3. Now that the coefficients of T_1, T_2 and T_3 are known we can define the coefficient of T_4.

The trace of a tour shown in Figure 2.21(b), shows that we can save 4 edges on the coboundaries of the handles by enabling a cycle to intersect the coboundary of T_4 in 4 edges. Two of these 4 edges are compensated for, by the extra 2 edges in the coboundary of T_3, the 2 others by the

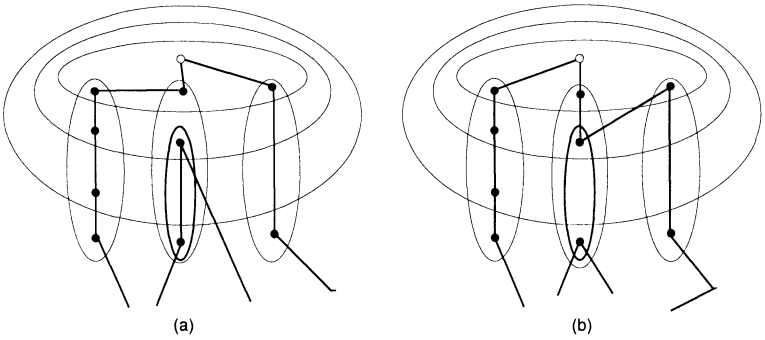

Figure 2.21. Some traces of Hamiltonian cycles

extra 2 edges in the coboundary of T_4, which yields a coefficient of $\beta_4 = 1$ for tooth T_4. This inequality is facet defining for $STSP(n)$, $n \geq 9$.

We use the developments of this section to describe a family of inequalities that contains the inequalities described in the previous sections.

10. A unifying family of inequalities

This section may be skipped by most readers. It would be too long to expose here the binested inequalities (see [613]). They contain the star (and therefore path), the clique tree, and the bipartition inequalities without degenerate teeth (in the sense of the previous section). They do not contain the ladder inequalities. One way of making them contain all remaining bipartition inequalities and the ladder inequalities would be as follows:

1 Recompute the coefficient of the teeth using the method of the previous section,

2 Arbitrarily order all edges which do not appear in a tight tour with the newly computed coefficients and compute sequentially the coefficient of the correcting term corresponding to them. The correcting coefficients will be subtracted to the LHS just like in the ladder inequalities.

We conjecture that the first calculation (the one for the teeth) is order independent. The second one is not. One can find an example in [615] in relation with generalizations of the ladder inequalities, for which this lifting is not order independent.

11. Domino inequalities

In the previous sections, teeth in some sense forced edges to cross the border of the handles or had to be no longer tight. This was obtained either because of parity or by the situation of nodes in key positions. There is another way of forcing edges in the border of the handles, that is by penalizing some edges, which must be taken in some circumstances if one does not cross the border of the handle. This is the rationale behind the *Domino Parity* inequalities defined by Letchford [559], although this is not explicit in his exposition. We describe here a large subclass of these inequalities which we will call *Domino* inequalities.

Let $H \subset V$, referred to as the *handle*, $\mathcal{T} = \{T_1, \ldots, T_i, \ldots, T_t\}$, $t \geq 3$ and odd, where $T_i = A_i \cup B_i \subset V$, referred to as the set of *teeth*, such that:

$$A_i \cap B_i = \emptyset \qquad \text{for } i = 1, ..., t \tag{55}$$

$$A_i \cup B_i \neq V \qquad \text{for } i = 1, ..., t \tag{56}$$

$$(A_i : B_i) \cap (A_j : B_j) = \emptyset \qquad \text{for } 1 \leq i < j \leq t \tag{57}$$

We say that A_i and B_i are the *two half dominoes* of the *domino* (or tooth) T_i. Note that there are no further restrictions on the elements of $\mathcal{T}$, i.e. they can for example intersect.

The main difference with the Domino Parity configurations of Letchford is that we added the non crossing Condition 57. Naddef proves in [614] that the only facet inducing Domino Parity configurations that do not satisfy this condition have two teeth, say T_1 and T_2 such that $T_1 \cup T_2 = V$ with $T_1 \cap T_2 \neq \emptyset$, but in that case there exists a domino inequality satisfying condition 57, which includes the same facet. Let $C = (\bigcup_{i=1}^{t}(A_i : B_i)) \setminus \delta(H)$ be the set of edges inside the teeth and crossing from one half domino to the other without crossing the boundary of the handle. The *domino inequality* is defined by:

$$x(\delta(H)) + \sum_{j=1}^{t} x(\delta(T_j)) + 2x(C) \geq 3t + 1 \tag{58}$$

Theorem 52 *Domino inequalities are valid for STSP(n).*

Proof: Call LHS the left hand sides of these inequalities. it is easy to check that:

$$LHS \geq \frac{1}{2}\sum_{i=1}^{t} x(\delta(A_i)) + \frac{1}{2}\sum_{i=1}^{t} x(\delta(B_i)) + \frac{1}{2}\sum_{i=1}^{t} x(\delta(T_i)) = 3t \tag{59}$$

Since LHS is even for all tours, we can raise $3t$ to $3t + 1$. ■

Note the similarity with the algebraic proof of validity of comb inequalities. To understand the role of the partition $\{A_i, B_i\}$ of each tooth, consider the example of Figure 2.22, where, except for T_7, $A_i = T_i \cap H$ and $B_i = T_i \setminus H$. For T_7, we have $A_7 \supseteq T_3 \cup T_4$, $B_7 \supseteq T_5 \cup T_6$. All teeth T_1 to T_6 are pairwise node disjoint. A tour for which all teeth are tight does not necessarily force more than six edges in the coboundary of H as seen in Figure 2.23(a), but then it necessarily uses an edge of $(A_i : B_i) \setminus \delta(H)$ which is *penalized* by two units. So either such a tour goes from A_i to B_i by using an edge of the coboundary of H as in Figure 2.23(b), or it uses a penalized edge to do so. In this example none of the other domino partitions play a role. The domino inequality corresponding to this example is facet inducing since it is a twisted comb inequality (definition below, see also [161]).

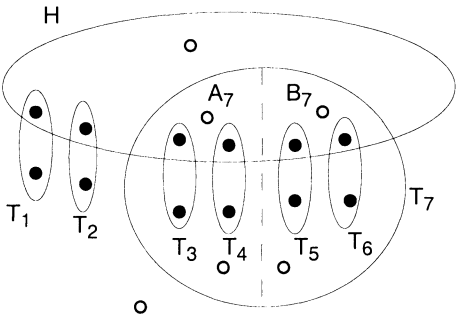

Figure 2.22. Example of domino inequality

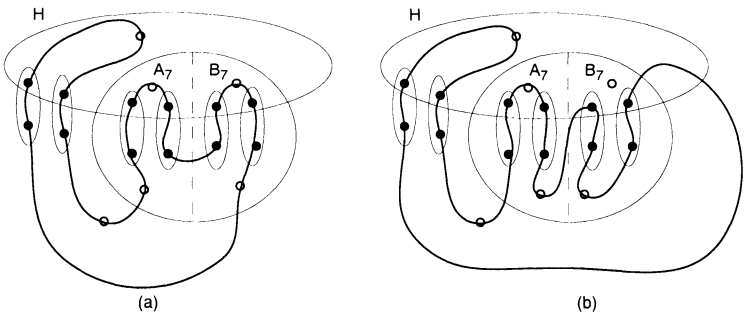

Figure 2.23. Tight tours for the example of Figure 2.22

We now turn to necessary conditions in order for a domino inequality to be facet inducing. A *minimal* domino configuration is one such that the number of non minimal (by inclusion) teeth is minimum, among all domino configurations that yield the same domino inequality. A comb

can be seen as a domino with only minimal teeth, but also as a domino configuration with one tooth containing all the others and such that one of its half dominoes contains all the teeth, the other half domino contains none, is non empty and has an empty intersection with the handle. Finally there is a node not in any tooth nor in the handle. Naddef in [614], proves the following propositions:

Proposition 53 *The teeth of a facet inducing minimal domino inequality are nested.*

Proposition 54 *If T_i is a minimal tooth in a facet defining domino inequality, then $\{A_i, B_i\} = \{T_i \cap H, T_i \setminus H\}$.*

Corollary 55 *Only the domino partition for non-minimal teeth has some interest. We will omit, from now on, giving the domino partition of minimal teeth.*

Minimal teeth are similar to comb teeth. For non minimal teeth, each tooth they contain must be contained in one of the half domino of that tooth. For minimal domino configurations it is shown in [614] that in a non minimal tooth each half domino must contain at least two *odd* teeth of immediate lower rank in the nesting. A tooth is odd if either it is minimal or it contains, including itself, an odd number of teeth.

Conjecture 56 *Minimal Domino inequalities with at least three odd maximal teeth and such that each half domino of a non minimal tooth contains at least two odd teeth, are facet inducing for STSPP(n).*

One can find in [614] the proof that a large subfamily of these inequalities is facet inducing for the graphical relaxation.

We end this section with a family of facet inducing domino inequalities. Boyd, Cockburn and Viella define in [134] the following *twisted combs*. A twisted comb inequality is a domino inequality satisfying all the previous necessary conditions and which has a unique maximal tooth which is not also minimal. Note that the example of Figure 2.22 refers to a twisted comb.

Theorem 57 *(Boyd, Cockburn, Vella [134]) The twisted comb inequalities are facet inducing for $STSPP(n)$.*

For more on the domino inequalities, besides the papers already mentioned, one can also read Cockburn [204].

Research Question 58 *How can one generalize the domino partition for non minimal teeth to inequalities with more than one handle?*

A first answer to this question can be found in [614] where a generalization of path inequalities can be found.

12. Other inequalities

There are many other inequalities which do not fall in one of the classes defined so far. The best known example are the *hypohamiltonian inequalities* (see [195]). A graph is said to be *hypohamiltonian* if it is *not* Hamiltonian but the deletion of *any* vertex yields a Hamiltonian graph. A hypohamiltonian graph is said to be *edge maximal* if it is hypohamiltonian and the addition of *any* non existing edge yields a Hamiltonian graph. The Petersen graph is an edge maximal hypohamiltonian graph.

Let $G = (V, E^*)$ be an edge maximal hypohamiltonian graph and $K_n = (V, E)$ be the complete graph on the same vertex set. The *simple hypohamiltonian* inequality $fx \geq f_0$ is given by:

1 $f_e = 1$ if $e \in E^*$

2 $f_e = 2$ if $e \in E \setminus E^*$

3 $f_0 = |V| + 1$

A general hypohamiltonian inequality is obtained by clique lifting of a simple hypohamiltonian inequality, that is replacing each vertex by a clique of some size, the coefficients of the inequality on the edges linking two nodes of the same clique is 0, the other coefficients are inherited from the ones of the simple inequality. In other words a hypohamiltonian inequality $fx \geq f_0$ is such that the partition of V into $V_1, \ldots, V_i, \ldots, V_p$, the vertex sets of the connected components of the graph $G_0 = (V, E_0)$ with $E_0 = \{e \in E : f_e = 0\}$ induces an edge maximal hypohamiltonian graph when one shrinks each V_i to a single node in $G_1 = (V, E_1)$, with $E_1 = \{e \in E : f_e = 1\}$.

Theorem 59 *Hypohamiltonian inequalities are facet inducing for* $GTSPP(n)$.

Proof: Let $fx \geq f_0 = p + 1$ be a hypohamiltonian inequality. Let the sets $V_1, \ldots, V_i, \ldots, V_p$ be the the partition of V induced by the connected components of $G_0 = (V, E_0)$. Inequality $fx \geq f_0$ is trivially valid since any closed walk must either contain an edge with coefficient 2, or two edges of coefficient 1 in some $(V_i : V_j)$, $i \neq j$. Therefore the coefficients sum up to at least $p + 1$ on the closed walk. Assume $fx \geq f_0$ is not facet inducing, i.e. the face it defines is strictly contained in some facet. Let $hx \geq f_0$ (same RHS), be the corresponding linear inequality. Every closed walk $W \in \mathcal{W}_f^=$ is also in $\mathcal{W}_h^=$. Let $W \in \mathcal{W}_f^=$. For $e \in E(V_i)$ for some i, we have $W + 2k\{e\} \in \mathcal{W}_f^=$ for all integer k, and therefore also in $\mathcal{W}_h^=$, which proves that $h_e = 0$ for these edges. As mentioned earlier, necessarily $h_e = \gamma_{ij}$ for all $e \in (V_i : V_j)$. Let W_i be a Hamiltonian cycle

of $G \setminus V_i$ that contains only edges of f value 0 or 1. Such a cycle exists since picking a single vertex in each V_i and the edges of f-value 1 linking them yields an hypohamiltonian graph. Let $e \in \delta(V_i)$ be such that $f_e = 1$. We have that $W_i + 2\{e\} \in \mathcal{W}_f^=$, so also in $\mathcal{W}_h^=$. this proves that all these edges have the same value $h_e = \gamma_i$. Since a hypohamiltonian graph is necessarily connected, we have that for $h_e = \gamma$ for all $e \in E$ such that $f_e = 1$, and therefore $\gamma = 1$, since we chose the same RHS. Now for any edge $e \in (V_i : V_j)$ such that $f_e = 2$, there is a Hamiltonian cycle containing it and only edges of f-value 0 and 1, which proves that $h_e = 2$ for these edges, and we have proved that $h_e = f_e$ for all $e \in E$. ■

We have described most of the known facet inducing inequalities that can be found in the literature, with the exception of the *crown* inequalities of Naddef and Rinaldi [617]. Most of the other known inequalities come from clique lifting of the inequalities obtained from the complete description of polytopes for small instances by Christof and Reinelt and which are available at: www.informatik.uni-heidelberg.de/groups/comopt/software/SMAPO/tsp/tsp.html.

13. The separation problem

The separation problem for the STSP is the following: *Given a fractional solution $x^\star$, find a valid inequality, preferably facet inducing for $STSPP(n)$, say $fx \geq f_0$, such that $fx^\star < f_0$ and such that $f_0 - fx^\star$ is not too small.*

We may search for a violated inequality in two ways. Either by searching among known classes of inequalities, for example among those we described in the previous sections, or in a very general form. Applegate et al. in [31] refer to the first type of search as a template paradigm separation. This has been for a long time the only type of STSP separation framework (see Padberg and Rinaldi [645], [647] and [648], Grötschel [397], Padberg and Crowder [229]). More recently Christof, Reinelt and Wenger have tried to use the complete knowledge of the $STSPP(n)$ for $n \leq 10$. We will give a brief description of their method in Section 15. Applegate, Bixby, Chvàtal and Cook [31] uses a more elaborate method which we will sketch in Section 18.

We end this section with the few complexity results known so far. Separating subtour elimination constraints is polynomial because it amounts to finding a minimum cut in an undirected weighted graph– we already addressed this point in Section 2. There also exists a polynomial time separation algorithm for *2-matching* constraints i.e. simple comb inequalities (see Padberg and Rao [644]).

Another result deals with bipartition inequalities with a fixed number of handles and teeth. Carr [170] and [171] showed that there is a polynomial separation algorithm in this case. We give the nice and elegant argument in the particular case of a comb with t teeth.

- Choose $2t$ vertices and, among these, choose t vertices that will belong to the handle. Match each of these t vertices of the handle to one of the t others to form t disjoint pairs.
- Find a minimum cut that separates the t nodes chosen to be in the handle from the t others. This can be done in polynomial time.
- For each pair of matched vertices, find a minimum cut that separates the pair from the other $t-2$ vertices. Choose one shore of the minimum cut to be a tooth.
- If the teeth intersect, there is an uncrossing argument that enables to obtain non intersecting teeth with the same coboundary values in $x^\star$.
- Check for violation.

Since there is only a polynomial number of choices for the $2t$ vertices, and for each such choice a polynomial number of choices thereafter, the whole procedure is polynomial for any fixed t. Of course this does not lead to a practical separation routine, but some of the separation routines that we will describe can be seen as trying to guess clever choices of some of these $2t$ vertices.

Fleischer and Tardos [310], designed a polynomial algorithm to separate maximally violated comb inequalities as long as the support graph of x^* (the graph induced by the edges e with ${x_e}^* > 0$) is planar. A comb is said to be maximally violated if the difference between the RHS and the LHS is 1 (.5 if in the standard "$\leq$" form).

In case of planar support graphs, Letchford [559] gives a polynomial algorithm that separates combs without the maximal violation requirement. This same algorithm also separates the domino inequalities under the same conditions.

The following has never been implemented and is so far of only a theoretical interest. Caprara, Fischetti and Letchford [161] address the separation of maximally violated cuts in the general context of ILP's of the form $\min\{cx : Ax \leq b, x \text{ integer}\}$, where A is an $m \times n$ integer matrix and b an m-dimensional integer vector. For any given integer k they study *mod-k* cuts of the form $\lambda Ax \leq \lfloor \lambda b \rfloor$ for any $\lambda \in \{0, 1/k, \ldots, (k-1)/k\}^m$ such that λA is integer. A mod-k cut is called maximally violated if it is violated by $(k-1)/k$ by the given fractional point x^*. It is

shown that, for any given k, separation of maximally violated mod-k cuts requires $O(mn \min\{m,n\})$ time. This result has applications to both the symmetric and asymmetric TSP. Indeed, for any given k, Caprara et al. propose an $O(|V|^2|E^*|)$-time exact separation algorithm for mod-k cuts which are maximally violated by a given fractional (symmetric or asymmetric) TSP solution with support graph $G^* = (V, E^*)$. This implies that one can identify in $O(|V|^2|E^*|)$ time a maximally violated mod-2 cut for the symmetric TSP whenever a maximally violated comb inequality exists. The reader is referred to [161] for more details.

We now turn to the heuristic separation of specific classes of inequalities.

14. Greedy heuristic for minimum cut

Finding a minimum cut that contains a given set S_0 is polynomial. In some of the following separation heuristics this search has to be performed so many times that we will instead use the following *greedy* heuristic – also known as *max-back*.

Consider a graph $G = (V, E)$ and a weight function x^* on the edges such that $x^*(\delta(v)) = 2$ for all $v \in V$. Let $S \subset V$, we say that $v \notin S$ *sees* S by an amount of $b(v) = \sum_{u \in S} x^*_{uv}$. We call $b(v)$ the *max-back* value of v relatively to S. A vertex not linked to S by any edge has a max-back value of zero. Note that this definition still makes sense if $v \in S$, and we will sometimes use it also in this case.

The max-back heuristic: Input $S_0 \subset V$ and x^*.

- $Cutmin = \sum_{e \in \delta(S_0)} x^*_e$, for all $v \notin S_0$, $b(v)$ = max-back value of v relative to S_0. Set $S = Smin = S_0$, $Cutval = Cutmin$

- While $S \neq V$: Choose $v \notin S$ of maximum max-back value. $S = S + \{v\}$, $Cutval = Cutval + 2 - 2 * b(v)$. For all $t \notin S$, set $b(t) = b(t) + x^*_{vt}$. If $Cutval < Cutmin$ then $Cutmin = Cutval$ and $Smin = S$.

The idea behind this heuristic is that if one wants to keep the cut small then one should take the vertices that see the current set by the largest amount. Note that the max-back idea is the key notion of the Nagamoshi-Ibaraki minimum cut algorithm (see [622],[623]). The terminology comes from A. Frank.

In general we will not explore the whole graph and will stop the previous loop after a certain number of iterations. We will also store the nodes not in S with a non zero max-back value in three lists:

- A list `supto1` which contains the nodes that see S by more than 1

- A list `infto1` which contains the nodes that see S by strictly less than 1.

- A list `equalto1` which contains the nodes that see S by exactly 1

These definitions may vary depending on the use we make of them. At this point we will just note that the insertion into S of nodes of the list `supto1` strictly decreases the value of the cut, and if not empty, the next chosen vertex is in that list. Those of `equalto1` will leave the value of the cut unchanged, whereas those of the last list will increase the value of the cut.

Some separation procedures use another greedy heuristic to find minimum cuts which we will describe later. In that heuristic a path is added at each iteration.

15. Graph associated to a vector $x^\star \in \mathbb{R}^E$

The *support graph* of $x^\star \in \mathbb{R}^E$ is the graph $G^\star = (V, E^\star)$ where $E^\star = \{e \in E : x^\star_e > 0\}$. Before performing separation it is useful to do some transformations on this graph.

By shrinking a set $S \subset V$, we mean replacing all vertices in S by a single one s, called a *pseudo-node*, and deleting all edges with both extremities in S. All the edges having exactly one extremity in S are replaced by edges whose extremity in S is replaced by s, the other is unchanged. This may lead to a multigraph, that is there may be more than one edge between a pair of vertices. It is convenient in that case to only deal with a single edge e whose associated weight $x^\star_e$ is the sum of all the weights of these edges. As the reader has observed, we will use the same notation, that is $x^\star$, for solution induced by $x^\star$ on the shrunk graph.

If one desires the degree inequality to be satisfied on the new node s only sets S with $x^\star(\delta(S)) = 2$ can be shrunk.

We say that the shrinking of a set $S \subset V$ is *legal* if every violated inequality in the the original graph yields such a inequality in the shrunk graph. This concept is very restrictive. Checking wether there is a violated inequality amounts to checking wether or not vector $x^\star$ is a convex combination of representative vectors of tours. We will come back to this later. Finally we define a shrinking *legal* for a class of inequalities, as a shrinking such that if the original graph contains a violated inequality of that class, then so does the shrunk graph.

Some easy recognizable cases of shrinkings are given in [647], two of them are given below. It is very important to perform these shrinkings before starting any separation heuristic, since they very often consid-

erably simplify the solution and reduce the size of the graph we work on.

A *path of 1-edges* is a maximal path P in the support graph $G^\star$ such that $x^\star_e = 1$ for all $e \in P$.

Most authors advocate replacing a path of 1-edges by a single edge, that is shrinking all the nodes except one extremity of the path. This shrinking is not legal in general for a given class of inequalities, but it is for comb separation. Naddef and Thienel [620] found useful to reduce only those paths with more than 4 edges and then shrink all nodes except the 4 vertices of the two extreme edges. This will leave a path of 4 edges, the middle vertex being the one to which all nodes have been shrunk into. The advantage is that this is legal for most classes of known inequalities and that if one uses the two extremities of one of the extreme edges as a tooth, then we know exactly what vertices we are dealing with.

The second shrinking concerns a path of 1-edges, the extremities u and v of which are adjacent to a common node w with $x^\star_{uw} + x^\star_{vw} = 1$. In this case we can shrink the node set S of the path. Note that this yields a new edge (s, w) with $x^\star_{sw} = 1$, which could yield another shrinking, therefore these shrinking operations have to be done recursively. Padberg and Rinaldi [647] show that this is a legal shrinking.

From now on we assume that these operations have been performed. An example of a graph obtained by such shrinkings is given in Figure 2.25 from the solution of Figure 2.24.

Christof, Reinelt and Wenger use the list of all minimum cuts to shrink the graph down to a graph on $k \leq 10$ nodes, in all possible ways, and look for a violated inequality in the list of inequalities describing $STSPP(n)$ for $n \leq 10$, see [184], [185], [187], [822] for more details.

16. Heuristics for Comb Separation

Comb separation has received a lot of attention. The reason is that comb inequalities are the first discovered and the simplest inequalities that do not appear in the integer formulation of the traveling salesman problem. We have already mentioned that the exact separation of the special case of 2-matching inequalities is polynomial.

It is useful to understand how a comb inequality can be violated by $x^\star$. Assuming that all subtour inequalities are satisfied by $x^\star$, each tooth has a coboundary of value at least 2. If each tooth has coboundary value 2, the previously mentioned proof of validity, shows that the coboundary of the handle cannot be less than $2k + 1$ and therefore maximum violation is 1 if all subtour elimination inequalities are satisfied. Therefore if the handle has a coboundary of $2k+2$ or more, no violated comb with $2k+1$

teeth can be violated. In general to have violation we must have that the sum of the excesses of the values of the coboundaries over their minimum values be less than 1. That is:

$$x^\star(\delta(H)) - (2k+1) + \sum_{i=1}^{2k+1} (x^\star(\delta(T_i)) - 2) < 1 \tag{60}$$

16.1. The biconnected component heuristic

A *cutnode* of a graph is a vertex whose removal disconnects the graph. If a graph does not contain any cutnode it is said to be *biconnected*. A *block* of a graph is a maximal biconnected induced subgraph. Finding all blocks is easy and can be done in linear time (see [787]).

Let $G^\star_{-1}$ be obtained from $G^\star$ by deleting all edges with $x^\star_e = 1$. Padberg and Rinaldi [647] use a heuristic in which each block or union of adjacent blocks of $G^\star_{-1}$ is considered as a potential handle H of a comb. The sets of extremities of an edge $e = (u, v)$ with $v \in H$, $u \notin H$, $x^\star_e = 1$ are used as teeth. The other teeth are chosen among the adjacent blocks, that is blocks which share a vertex with H. We advocate to try to grow by the max-back procedure a tooth from each cutnode of $G^\star_{-1}$ contained in H. This procedure is quite effective in the beginning of the Branch-and-Cut procedure. Moreover, it is very fast.

Note that if the paths of edges of value 1 have not been contracted to a unique edge, then one must add to each block the nodes of such paths linking two of its nodes.

Figure 2.24 gives an example of a fractional point of kroA100, the dotted edges correspond to value .5, the others to value 1. Figure 2.25 gives that same graph after contraction following the rules described earlier. The blocks happen here to be exactly the connected components of the graph from which the edges e with $x^\star_e = 1$ have been removed and correspond to the nodes of the 6 triangles. Note that in the original graph there was one more block. Several violated 2-matching inequalities are easily found, some in the original graph are comb inequalities which are not 2-matching inequalities. Figure 2.26 shows two of these inequalities and one of the four 2-matching inequalities.

16.2. The max-back Heuristic

In this heuristic, starting from a given node chosen in a certain subset B, we grow a handle using a modified max-back procedure. Once in a while we stop growing the handle and try to grow a number of teeth using again a max-back procedure. If a violated comb has been obtained we stop, else we go back to grow the handle, and so on as long as no

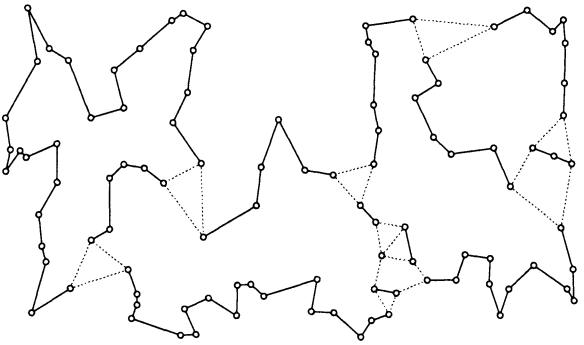

Figure 2.24. Example of a fractional point

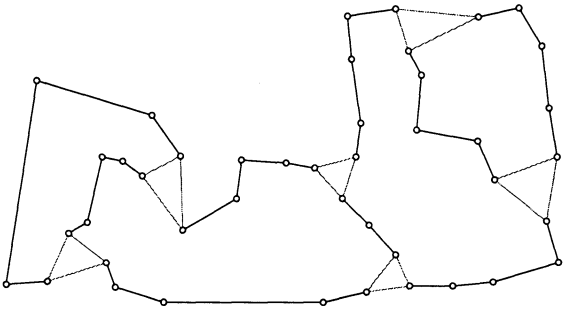

Figure 2.25. The graph of Figure 2.24 after shrinking

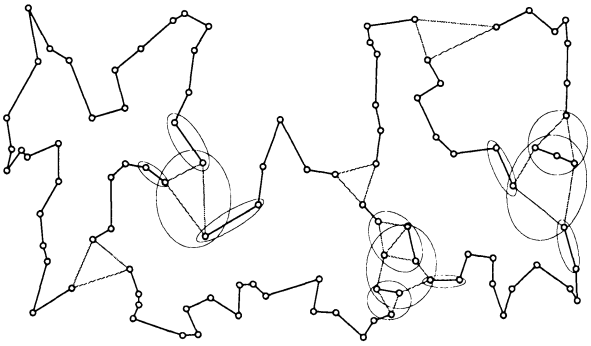

Figure 2.26. Examples of violated combs from Figure 2.24

violated comb has been found and the maximum number of iterations

has not been exceeded. We then choose another node from B until we have tried with all nodes in B. We make this more precise now.

In our case we chose B to be all extremities of paths of 1-edges. Let $u \in B$, we grow S by max-back starting with $S_0 = \{u\}$. Let v be such that $x^\star_{uv} = 1$, the idea is that $\{u, v\}$ will be the first tooth. One therefore forbids the max-back procedure to pick up any other node of the path of 1-edges one extremity of which is u. Nodes of max-back value at least one and not incident to a 1-edge linking it to the current growing handle are included in the handle under construction. Those nodes incident to a 1-edge are chosen if their max-back value is greater than a prescribed value (say 1.7). The idea is that we lose a potential "nice" tooth, but on the other hand the coboundary of the handle will decrease in value. Whenever we introduce a node v in the handle, we also update the max-back value of all the nodes adjacent to it, including those which are already in the handle under construction.

Every time the previous procedure would choose a node of max-back value strictly less than 1 and the coboundary value is not too close to an even number, we first try to find a violated comb by searching for the right number of teeth. This number is easily computed knowing the value of the coboundary of the current handle. If that value is strictly between $2k$ and $2k + 2$, we need $2k + 1$ teeth. One tooth is already known, as already mentioned. Nodes of the handle which are incident to a 1-edge the other extremity of which is not in the handle, give us additional teeth. For the remaining teeth, we greedily choose the vertex in the handle with minimum max-back value among those not already in a tooth as the starting node for the next tooth. The rationale behind this is that if a node is poorly linked to the other nodes of the handle, it is likely that we will find a set with small coboundary value containing it that will yield a tooth. The tooth is grown by pure max-back from that node, leaving it grow freely inside or outside the handle, except that it cannot intersect already existing teeth. We do so until either we reach a value very close to 2, or a maximum number of iteration has been exceeded. We then return the best set found. In fact, in view of the search for other inequalities, we return more sets than just the best one, especially if that set contains only one node outside the handle. For other inequalities we will need the information on the best tooth having at least two vertices outside the handle. The weakness of this procedure is that we never change a previously found tooth, hence depending on the order in which we consider the starting nodes, we may or may not succeed in finding a violated comb. This is however unavoidable due to the nature of the greedy heuristics. In the code of Naddef and Thienel [620], the minimum max-back value for the choice of the starting node

of a tooth suffers many exceptions, the main one is as follows. If a path of 1-edges linking say u and v is entirely inside the handle, it is very often the case that a useful tooth will contain the whole path. In this case we consider that u and v as having a max-back value equal to the sum of their real max-back value minus two.

This heuristic is fast, pretty effective and easy to implement. It suffers from the fact that in many instances many max-back values are identical and one has to be lucky to choose the right node. This is especially true at the beginning of the cutting phase procedure. This is why Naddef and Thienel prefer the much slower but much more successful heuristic described in the next section.

Note that, in some way, this relates to Carr's comb separation algorithm. Once the handle is chosen, we choose the nodes in the handle which will be the seeds, as Carr calls them, of the teeth. The difference is that Carr's method also chooses one seed outside the handle for each tooth.

16.3. The Ear Heuristic

We now give another greedy way of growing sets. Given $S \subset V$, an *ear* relative to S is a (possibly closed) path in G with intermediate nodes not in S. In most graph theoretic domains where ears are used, i.e. connectivity or matching theory, a single edge between two nodes of S is considered as an ear, and this path need not be minimal in terms of vertices outside S. Here we will assume that there is at least one node not in S in the path and that, except for the first and last node of the path outside S, none of the nodes of that path are incident to nodes of S.

Given $S \subset V$, by "increasing S by an ear", we mean adding to S the ear that yields a set of minimum coboundary value among all sets that can be obtained this way. Of course, like in the case of the max-back procedure, when growing handles we are faced with the dilemma of choosing or not nodes linked to S by a 1-edge. We use the same rule as in the previous case. In fact the only difference with the previous section is the way sets are grown. In some way we control better the way the set grows and therefore this method typically gives much better results. The only problem is that one has not yet found an efficient method to find the best ear: if we could improve this part substantially, the running times reported below using the Naddef and Thienel separation heuristics would be drastically decreased.

16.4. About minimum cuts

The two comb separation heuristics we will describe later on, as well as the exact separation algorithms for violated combs and domino inequalities in planar graphs we have mentioned, make extensive use of the structure of minimum cuts. Let us assume that all subtour elimination are satisfied by $x^\star$, that is $x^\star(\delta(S)) \geq 2$ for all $S \subset V$. Therefore the value of a minimum cut is 2, since each node defines such a cut. Finding all cuts of value 2 is relatively easy. There are several ways of storing all these cuts in a compact form, either by a *cactus* representation of Lemonosov, Karzanov and Timofeev (see for example [310], [309], [241], [822]), by a PQ-tree of Booth and Lueker [131] or by a poset representation (see [342]).

The main property of minimum cuts which is used is that if the sets S_1 and S_2 define minimum *crossing cuts*, that is $S_1 \cup S_2 \neq V$, $S_1 \cap S_2 \neq \emptyset$, $S_1 \setminus S_2 \neq \emptyset$ and $S_2 \setminus S_1 \neq \emptyset$, then $S_1 \cup S_2$, $S_1 \cap S_2$, $S_1 \setminus S_2$ and $S_2 \setminus S_1$ all define minimum cuts. In particular, $x^\star$ representing the capacities of the edges, we have $x^\star(S_1 : V \setminus (S_1 \cup S_2)) = x^\star(S_1 \setminus S_2 : S_1 \cap S_2) = x^\star(S_2 : V \setminus (S_1 \cup S_2)) = x^\star(S_2 \setminus S_1 : S_1 \cap S_2) = 1$, $x^\star(S_1 \cap S_2 : V \setminus (S_1 \cup S_2)) = 0$ and $x^\star(S_1 \setminus S_2 : S_2 \setminus S_1) = 0$. Figure 2.27 summarizes this.

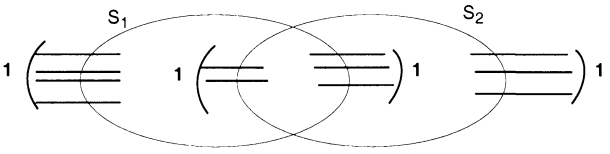

Figure 2.27. Two crossing minimum cuts

We only briefly describe here the PQ-tree structure to store minimum cuts. For each cut the shore that does not contain a prescribed node a is stored. A *PQ-tree* is a rooted tree with each internal node having at least two children and labelled either P-node or Q-node. Following [28], for a node u of a PQ-tree, we let $D(u)$ be the set of leaves of that tree that are descendants of u.

A PQ-tree represents all the shores of the minimum cuts not containing node a if and only if each shore is either:

(i) $D(u)$ for some node u of the PQ-tree

(ii) The union of $D(u)$s for *consecutive* sons of a Q-node

(iii) The union of $D(u)$s for any subset of sons of a P-node

and each such set is the shore of a minimum cut. Note that the leaves are exactly the nodes of $G \setminus \{a\}$. Without going into the details of the

PQ-tree structure, from the preceding lines it is obvious that the sons of a Q-node are not placed in an arbitrary order.

16.5. The Domino Heuristic

Assuming all subtour elimination inequalities are satisfied, a $2k+1$ tooth comb is maximally violated if the handle has a coboundary value of $2k+1$ and each tooth is tight. That is, all teeth define minimum cuts, but also so do the intersection of each one with the handle and its intersection with the complement of the handle. Moreover for each tooth T_i we have: $x^\star(T_i \cap H : H) = x^\star(T_i \cap H : T_i \setminus H) = x^\star(T_i \setminus H : V \setminus H) = 1$ and therefore $x^\star(H : V \setminus (H \cup_{i=1}^{2k+1} T_i)) = 0$; see Figure 2.28 where the only edges that can cross the boundary of one of the sets that define the comb are shown.

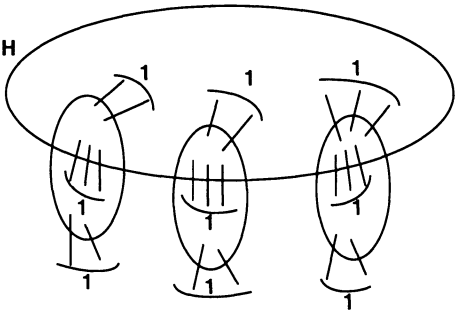

Figure 2.28. Edges in maximally violated comb

Applegate et al. [28] call a set $T = A \cup B$, such that $A \cap B = \emptyset$ and $x^\star(\delta(T)) = x^\star(\delta(A)) = x^\star(\delta(B)) = 2$ a *domino*. We call an edge $e \in (A : B)$ such that $x_e^\star > 0$ a *crossing edge* of the domino $T = A \cup B$.

If we find a set of $2k+1$ pairwise node disjoint dominoes such that the union of their crossing edges form a cut of the support graph of $x^\star$, then we have found a maximally violated comb. The handle is one of the shores of that cut. Such a set is called a *cutter* in [28]. This is the idea behind the heuristic proposed by Applegate et al. [28].

A *necklace* is a partition $V_0, \ldots, V_i, \ldots, V_k$ of V into tight sets such that $S = (V_i \cup V_{i+1})$ forms a domino for all i, where subscripts are taken modulo k. Each Q-node v of a PQ-tree compatible with $x^\star$ defines a necklace with $V_0 = V \setminus D(v)$ and $V_i = D(v_i)$ where $v_1, \ldots, v_i, \ldots, v_k$ are the sons of v in that order.

Given a PQ-tree compatible with $x^\star$, let $\mathcal{D}$ be the set of all dominoes of all the necklaces obtained from the Q-node of that tree. Applegate et al. in [28] search $\mathcal{D}$ in a clever way in order to find a set of dominoes

which defines a cutter and therefore yields a violated comb. We refer the reader to [28] for more details. It very often happens that the support graph of a fractional solution is planar and therefore, one may now rather try to use the newly available algorithms of Fleischer and Tardos [310], of Letchford [559] or of Caprara, Fischetti and Letchford [161], although this latter has, so far, not yet been implemented.

16.6. The Consecutive Ones Heuristic

We just give here a flavor of the consecutive ones heuristic of Applegate et al [28]. It is easier to understand if one assumes that the minimum cut has a value of two, and that we have a PQ-tree that represents the set of all minimum cuts. This is not really a restriction since some TSP codes as the one of Naddef and Thienel [620] and [621], only separate on other constraints when all subtour elimination inequalities are satisfied.

A set of subsets of a given ground set is said to have the *consecutive ones property* if one can number the elements of the ground set in such a way that all the elements of any subset have consecutive numbers. Given $S_i \subset V \setminus \{a\}$, $i = 1, \ldots, k$, where a is as prescribed node as in Section 16.4, if all these subsets are tight for a certain tour Γ, then they satisfy the consecutive ones property since numbering consecutively the nodes in the order induced by Γ yields the desired property.

Let H be a vertex set such that $x^\star(\delta(H)) < 4$. Any tour Γ either intersects the coboundary of H in two or at least four edges. Let's try and see if it is possible in two edges, that is we want to test whether H can be tight, when all the currently tight sets remain tight. There is an algorithm that enables to refine the current PQ-tree to incorporate the cut defined by H or returns a failure message together with a certificate explaining why this is not possible. This certificate has the form of three tight sets T_1, T_2, T_3 which are incompatible with the fact that H could be added to our set of subsets preserving the consecutive ones property. The sets H, T_1, T_2, T_3 define a violated comb inequality since the three sets T_1, T_2, T_3 are either pairwise disjoint or one contains the two others, which do not intersect.

The problem is the choice of the set H. It could be the handle of a former violated comb, plus or minus some nodes. That is, one starts with the handle of a former comb, and possibly increase it by the max-back procedure described earlier.

Research Question 60 *Assume the set of minimum cuts is stored in a cactus instead of a PQ-tree. Can we obtain a certificate that a given*

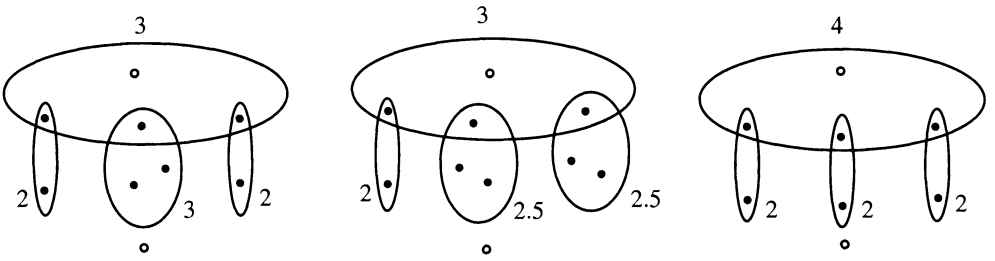

Figure 2.29. Examples of non violated comb inequalities

set cannot define a minimum cut if all the current minimum cuts remain minimum? Same question if one uses a poset representation.

To make the question more precise: Take a set S which does not define a minimum cut. Assume the solution $x^\star$ has changed in such a way that all the minimum cuts remain minimum. Can S now also define a minimum cut? In case of a NO answer a good certificate is asked for.

17. Separation of multi-handle inequalities

17.1. How do multiple handles help?

Most of the non-comb inequalities for $STSPP(n)$ we have described have combs as building blocks and therefore have more than one handle. If one wants to design specific separation heuristics (in a template paradigm) it is important to understand how the addition of a handle can yield a violation when none of the constituting comb inequalities is violated. This is explained in details in Naddef and Thienel [621]. Figure 2.29 shows non violated (in fact, tight) comb inequalities. These could have been found in our search for violated combs.

All the figures of this section will follow the following convention as far as numbers are concerned: cut values will be given in floating point format, i.e. 2.0, 3.2, to distinguish them from set coefficients which are integers. In some of the forthcoming figures we also have drawn in bold the non violated comb inequality from which we have started the search for a multi-handle violated inequality.

The key remark in understanding how one of these inequalities can be violated, is to see that if a tooth with large cut value intersects *properly* enough handles with cut value close to an odd integer, then it is likely to be part of a violated inequality. We illustrate this in the following figures.

Figure 2.30 shows how violated path inequalities can be found on the two first cases of Figure 2.29, as long as one can find a second handle that

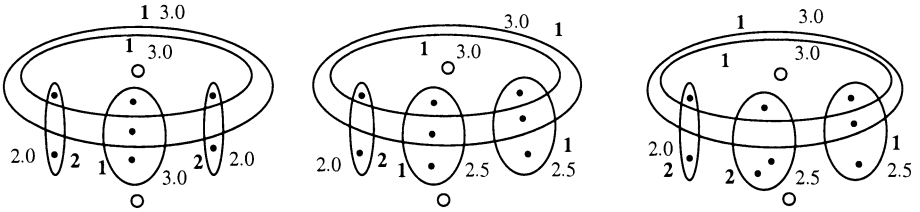

Figure 2.30. Examples of violated path inequalities with 2 handles

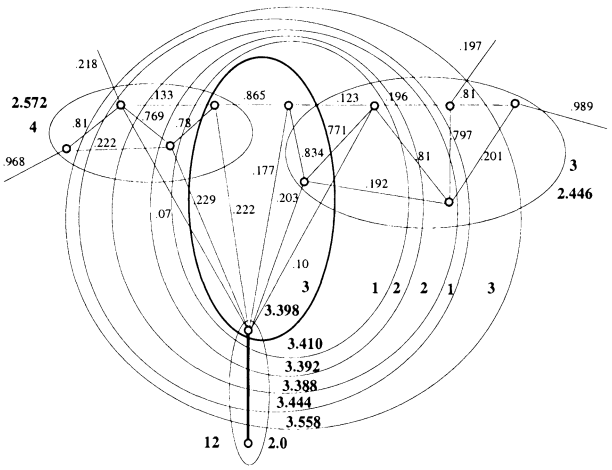

Figure 2.31. Example of a violated path inequality in ts225

intersects the teeth as shown (this is what we meant by *properly*). The last of these drawings shows a case in which the second handle intersects one of the teeth in such a way that the tooth gets a higher coefficient, yielding a weaker inequality. In the two first drawings, the inequalities are violated by 1, it is violated by only .5 in the last one. Note that teeth with cut value close to 2.0 are penalized only marginally if they receive higher coefficients. Figure 2.31 shows an example on the instance ts225, of a non violated comb inequality (central handle in bold) and a violated path inequality with the same set of teeth. The RHS is 76 and the LHS 70.3.

The same argument on teeth with high cut value holds if the handles are disjoint instead of being nested. This yields either bipartition or clique tree inequalities. Figure 2.32 refers to the case of clique trees if only one tooth has a large cut value, and Figure 2.33 to the case in which more than a tooth has such a cut value.

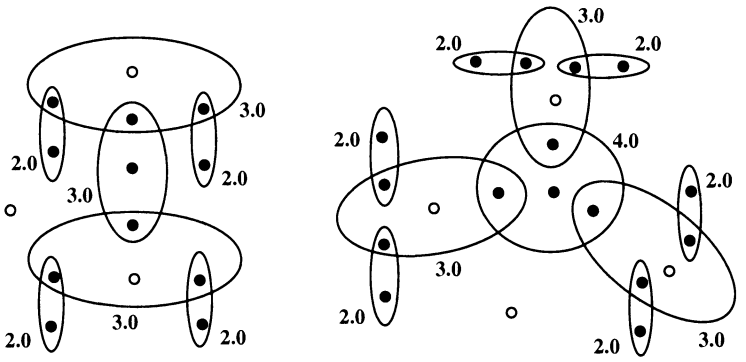

Figure 2.32. Examples of violated clique trees

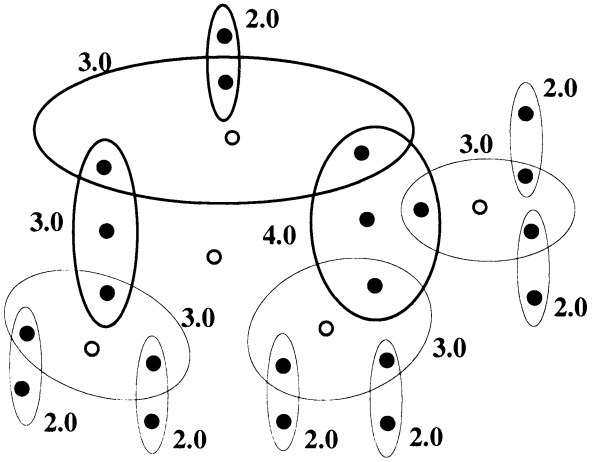

Figure 2.33. Example of violated clique tree

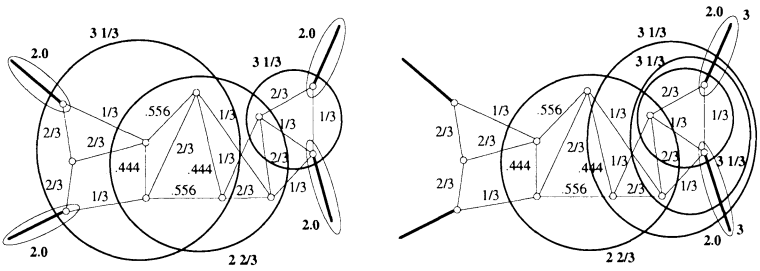

Figure 2.34. Violated clique tree and path inequalities from pr299

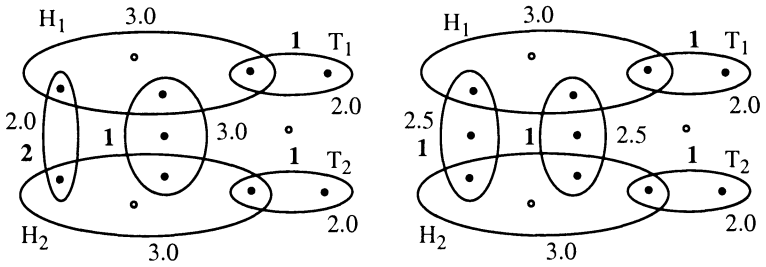

Figure 2.35. Example how ladders can be violated

Figure 2.34 gives an example of a clique tree inequality that can be found this way. It is taken from pr299. The RHS is 18 and the LHS 17.333. In this case there is also a violated path inequality, as shown in the second part of the figure; the two teeth made up by an edge have coefficient 3, the RHS 26 is and the LHS is 24.667. The path inequalities defined by any two of these three handles is also violated.

Ladder inequalities may also help in the same way, but since there can only have two handles, one cannot expect them to accept too high cut values for the teeth. Using the handles to compensate for high cut values for the teeth adds up in using a ladder inequality as a bipartition inequality on two handles (see Figure 2.35). We will see later on how ladders can help to compensate for bad cut values of handles, that is values too close to an even number. The correcting term will then be the key factor. Figure 2.36 gives an example of ladder violation in gil262, the RHS is 16, the LHS is 15. Figure 2.37 gives two violated ladders in a fractional solution to pr439, one with disjoint handles, the other with nested handles, in both cases the RHS is 18, the LHS is 17.667.

Note that so far we have shown how to deal with cases related to combs of the two first types of Figure 2.29, that is in the case in which non-violation is due to teeth of too high cut value. In the last case of that figure, the handle has a too high cut value and finding extra handles

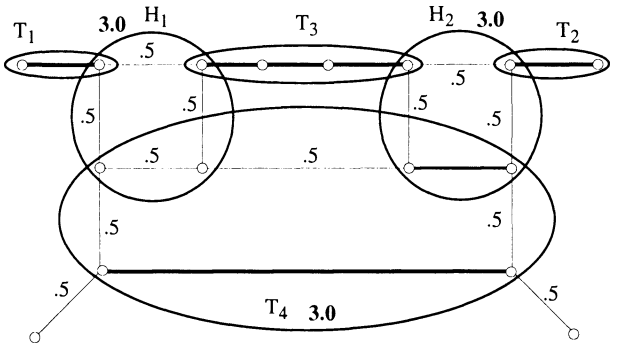

Figure 2.36. Example of ladder violation in gil262

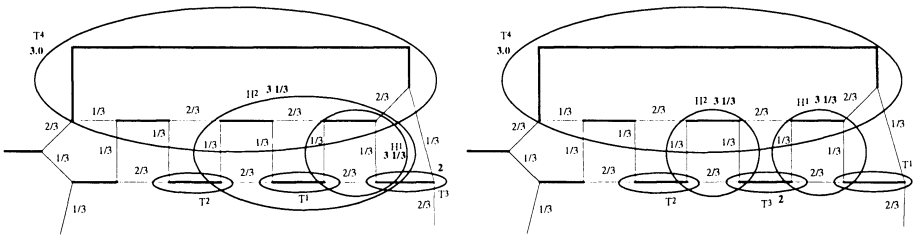

Figure 2.37. Example of violated ladders from pr439

in a way or another would not help. The only known possibility is to use the correcting term of the ladder inequalities. Figure 2.38 shows how this can happen on an example from a fractional solution to pr136. The first drawing shows a tight comb with a handle of cut value 4, the second shows a violated ladder in which the violation comes from the correcting term induced by the edge with black filled extremities. Figure 2.39 gives two examples taken from two fractional solutions to pr439 in which the handles are nested. In both cases the violation comes from the correcting term given by the edge, the two extremities of which are filled in black. All combs that can be built using one of the handles and the three teeth it intersects are not violated. The violation in the first case is 2/3 and 0.518 in the second.

How can we hope to find such violated multi-handle inequalities? Naddef and Thienel [621] suggest the following strategy.

When searching for teeth in combs, if the best found tooth only has one node outside the handle and its cut value is not close to 2, also return the best tooth with at least two nodes outside the handle. For example, in Figure 2.31 the edge of value .78 in the coboundary of the inner handle yields a tooth of cut value 2.44 which is better than the 2.57

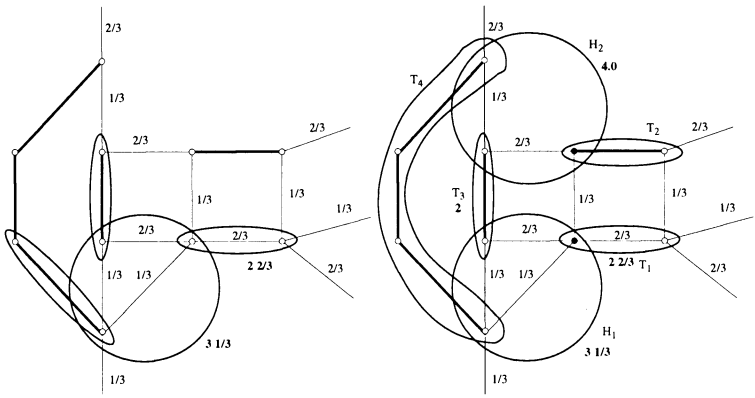

Figure 2.38. Violated ladder from pr136 with correcting term

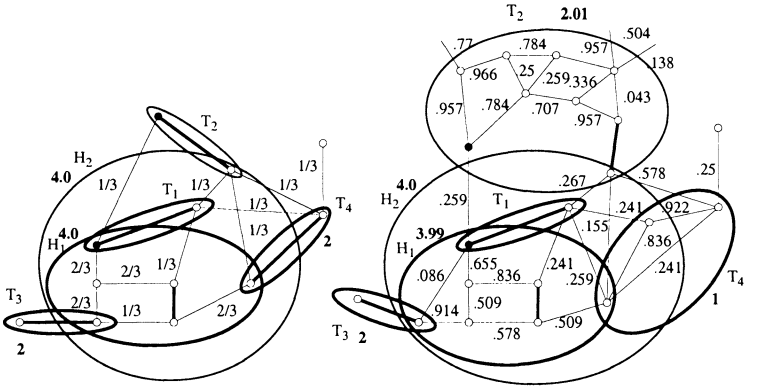

Figure 2.39. Violated ladders from pr439 with correcting term

cut value of the chosen tooth. This is not computationally expensive, since in the search for a tooth one goes anyway beyond the set which is finally returned. The idea is that a tooth with large cut value having only one node outside the handle will be a handicap towards violation that cannot be compensated by finding extra handles as shown in the previous section.

The general strategy is based on a non-violated comb. If we have large enough teeth, in terms of number of nodes outside the handle, we search for a violated path inequality. We try to grow handles, starting with the current one such that their cut values are low enough and that they yield small coefficients for the teeth with large cut value. We can also try to find handles included in the previous one. All this is done by the max-back or the ear procedure described earlier.

If one does not succeed, then one tries to find another type of violated inequality, either in a way similar to that for paths (ladder inequalities with nested handle) or by first growing a new handle which intersects one or several teeth of large cut value, and then searching for new teeth and so on. Everything is done in a greedy way. These simple greedy procedures give surprisingly good results as will be reported in the section on computational results. The interested reader is referred to [620] and [621] for more details.

18. Separation outside the template paradigm

So far we have developed separation heuristics that try to find a violated inequality from a prescribed family of valid inequalities. Some other paths have been explored. Applegate, Bixby, Chvátal and Cook in [30] and in [31] have explored another direction which is the topic of this section.

Figure 2.40 shows a portion of a fractional point of the instance fnl4461. When developing a STSP solver, one very often looks at such drawing of a fractional solution. The fact that one can quite easily visualize fractional solutions, is certainly one of the keys to the progress that has been possible for this particular problem.

When doing so, one looks at small parts of the drawing to try and figure out what one has missed in terms of violated inequalities. In other words, one locally searches for violated inequalities. Therefore we will refer to the following procedure as the *local cut* procedure (which in fact was the first name their authors gave it). For example one may wonder if there is a violated inequality involving the black filled nodes of Figure 2.40. We will see that the answer "yes" will be very easy to

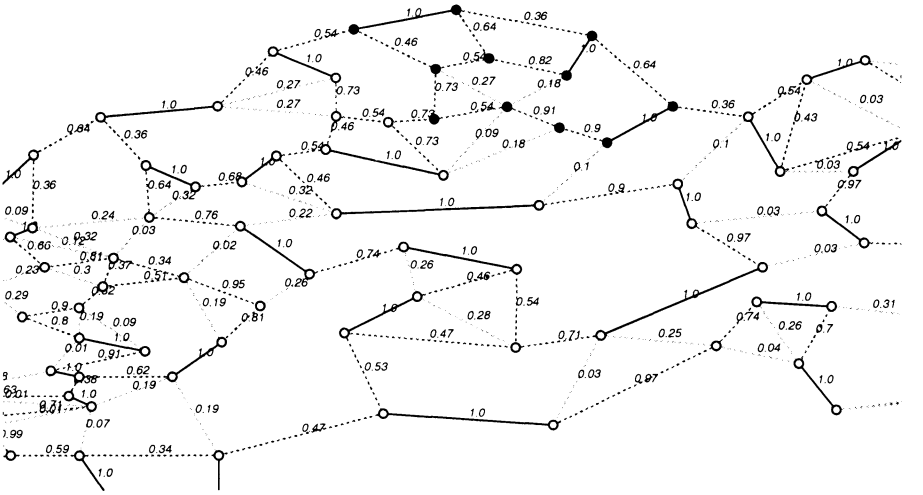

Figure 2.40. Part of a fractional solution of fnl4461

find. Much harder will be the answer to the question of knowing which valid inequality is violated.

Let us contract all the other nodes into a single one to obtain the graph of Figure 2.41 in which the larger circled node represents the node obtained from the shrinking. We will refer to that node as the *pseudo-node*. Note that in both figures, all nodes may in fact represent more than a single node in the original problem because we have applied the traditional legal graph reductions described earlier. This will only matter at the end of our computations.

It happens here that the sum of the $x_e^\star$ on the star of the pseudo-node is two. If we impose this condition, then the *local search* for violated inequalities may turn short for lack of interesting sets. This is why in their development, Applegate et al. [31], do not require this condition. If that value is less than two we have a violated subtour elimination inequality, else it could be anything greater or equal to two. To make things easier to expose, we will assume that this sum is two and try to adapt their method to this case. Let $\tilde{G} = (\tilde{V}, \tilde{E})$ be the small graph obtained after shrinking a very large set of nodes into a single pseudo node. We continue calling $x^\star$ the fractional solution induced on it by our fractional vector $x^\star$.

If in $\tilde{G}$ the solution $x^\star$ (see Figure 2.41) is a convex combination of incidence vectors of Hamiltonian cycles, then $x^\star$ violates no valid inequality in $\tilde{G}$. Else there is necessarily at least one such inequality.

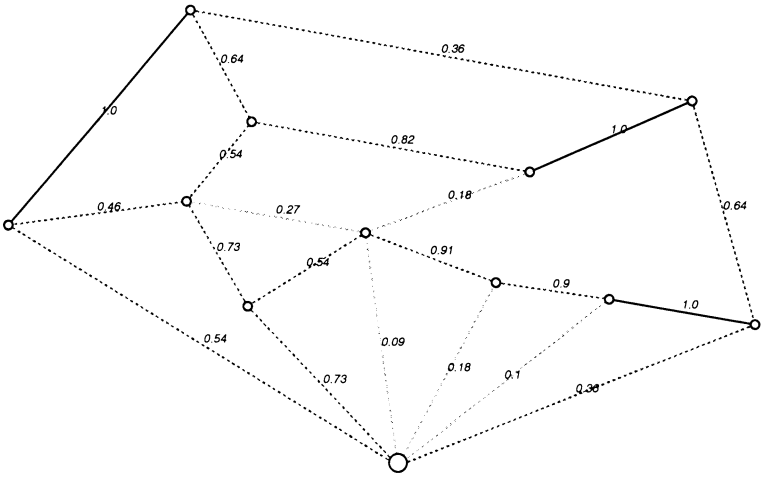

Figure 2.41. The same solution after shrinking

Note that any Hamiltonian cycle candidate to be used in such a convex combination must contain all edges e such that $x_e^\star = 1$ and no edge for which $x_e^\star = 0$. We call such tours *compatible* with $x^\star$. In our example there are only four possible such Hamiltonian cycles. None uses the edge of value .18 not incident to the pseudo node, and therefore this solution is not a convex combination of incidence vectors of Hamiltonian cycles and it is worthwhile to search for a violated inequality in this graph.

To make things more precise, one can solve the following linear program which tries to write $x^\star$ as a convex combination of incidence vectors of Hamiltonian cycles. The summations are on all compatible Hamiltonian cycles Γ except when expressed differently:

$$max \quad \sum_\Gamma 0\lambda_\Gamma \tag{61}$$

$$\sum_{\{\Gamma: e \in \Gamma\}} \lambda_\Gamma \quad = \quad x_e^\star \text{ for all } e \in \tilde{E} \tag{62}$$

$$\sum_\Gamma \lambda_\Gamma \quad = \quad 1 \tag{63}$$

$$\lambda_\Gamma \quad \geq \quad 0 \text{ for all } \Gamma \tag{64}$$

This problem can be solved by generating first all compatible Hamiltonian cycles, which in general are in small number if the number of nodes is small and we have some edges e with $x_e^\star = 1$, since these restrict a lot the number of choices. Note that $\tilde{G}$ is in general not dense since,

as we will see in the next section, we work on a very restricted set of edges (variables). This problem can be also solved by dynamic column generation. Each column generation is a TSP problem using the dual variables as costs, but on a small instance with many fixed variables. We will comment on this point later on.

If the above LP has a solution then we are out of business, else by the Farkas lemma there exists an inequality $fx \leq f_0$ which explains this infeasibility, i.e. such that $fx \leq f_0$ for all compatible tours and $fx^\star > f_0$. Some commercial linear programming software, such as CPLEX, provide a function that return the coefficients f_e for $e \in \tilde{E}$ of such an inequality.

In [31] the following slightly different linear program is solved in order to get the coefficients f_e directly from the dual without having to use an ad hoc procedure. Here again Γ is a generic compatible tour.

$$max \quad s \tag{65}$$

$$-\sum_{\{\Gamma: e \in \Gamma\}} \lambda_\Gamma + s x^\star_e + w_e = 0 \text{ for all } e \in \tilde{E} \tag{66}$$

$$\sum_{\Gamma} \lambda_\Gamma - s = 0 \tag{67}$$

$$-1 \leq w_e \leq 1 \text{ for all } e \in \tilde{E} \tag{68}$$

$$\lambda_\Gamma \geq 0 \text{ for all } \Gamma \tag{69}$$

This linear program is unbounded if and only if $x^\star$ is a convex combination of incidence vectors of tours of $\tilde{G}$. Else letting f_e, for $e \in \tilde{E}$ be the dual variables associated to constraints (66), f_0 that associated to constraint (67), u_e and v_e those associated to the bounds on the components of w, the dual writes:

$$min \quad \textstyle\sum_{e \in \tilde{E}} (u_e + v_e) \tag{70}$$

$$-\sum_{e \in \Gamma} f_e + f_0 \geq 0 \quad \text{for all compatible tour } \Gamma \tag{71}$$

$$\sum_{e \in \tilde{E}} f_e x^\star_e - f_0 = 1 \tag{72}$$

$$f_e + u_e - v_e = 0 \quad \text{for all } e \in \tilde{E} \tag{73}$$

$$u_e \geq 0, v_e \geq 0 \quad \text{for all } e \in \tilde{E} \tag{74}$$

Any feasible solution to this dual problem yields an inequality $fx \leq f_0$ which is valid for all compatible tours as shown by constraints (71) and such that $fx^\star = f_0 + 1 > f_0$ by Constraint (72). Note that due to the Objective Function (70), if one finds a violated inequality, one finds

one which minimizes $\sum_{e \in \tilde{E}} |f_e|$, so it is in some sense a most violated inequality.

The steps in [31] (slightly modified for sake of simplicity to the case of a pseudo-node of "degree" two) are:

1 Turn the inequality $fx \leq f_0$ into one with integer coefficients.

2 Transform it into an inequality valid for all tours which only use edges in $\tilde{E}$.

3 Transform it into a facet inducing inequality on $\tilde{G}$.

4 Perform a sequential lifting to compute the coefficients on the edges not in $\tilde{E}$. Assume one has transformed the inequality into one in "greater or equal" form, $gx \geq g_0$, which can easily be done by multiplying by -1 both sides and then adding some degree equalities in order to bring back all coefficients non negative. Then each lifting operation refers again to a STSP on a very small graph, which does not mean that things will be necessarily easy. Experiments carried out by Jünger, Reinelt and Rinaldi [474] show that when the coefficients of the objective function are those of weird facet inducing inequalities, then the TSP may be extremely difficult to solve by Branch and Cut even for very small instances. For this last reason it is advisable to put a time limit on this phase and return a failure message if it exceeds that time.

5 Transform the inequality into a facet inducing inequality using a tilting procedure inspired by Minkowski. Procedure also used in Step (2). Note that Applegate et al. do not need this step, but when dealing with Hamiltonian cycles lifting an edge may increase the dimension of the underlying polytope by more than one unit, which is not the case when one uses closed walks. On the other hand, Applegate et al. need a more sophisticated routine for the column generations and the lifting procedures which are no longer TSPs.

6 Put in TT form. For technical reasons due to their Branch-and-Cut code, Applegate et al. only keep those TT inequalities that can be put in closed set form.

7 Finally the shrunk nodes are expanded using 0-liftings.

Applegate et al. report excellent results with their procedure. We will report some of their results in the section devoted to computational results.

19. Branch-and-Cut implementation of the STSP

The Branch-and-Cut method is very simple in its principle, but once we want to use it there are many technical points to address. Figure 2.42, taken from Jünger, Reinelt and Rinaldi [474], shows the flowchart of a Branch-and-Cut implementation. For a detailed study on Branch-and-Cut we refer the reader to Jünger and Thienel [477], [478], Thienel [790], Jünger, Reinelt and Rinaldi [474]. From now on *lp* stands for linear program. We will only address the following points:

- Management of variables

  active variables: What variables should one work with?

  pricing: How often should we apply variable pricing, and how should one do it?

  fixing: How to perform variable fixing?

- Management of constraints

  separation: What separation strategy should one choose?

  storing: How should one store the constraints?

  cleaning: How to keep the lp of reasonable size?

- How to obtain a good feasible solution?

- Branching

  When should one decide to stop separating and start branching?

  How should we perform the branching and how much effort should one invest in it?

  Which subproblem has to be solved next?

We will have to refer very often to two implementations of the Branch-and-Cut algorithm for the STSP, that of Applegate, Bixby, Chvàtal and Cook, and that of Jünger, Naddef, Reinelt and Thienel. We will refer to them by ABCC and JNRT.

19.1. Variables

Since we are working on complete graphs it is out of question to have all the variables present in the lps. One starts with a reasonably sized set of variables. The set of corresponding edges is usually referred to as the edge set of the *sparse graph*. There are various ways of choosing the

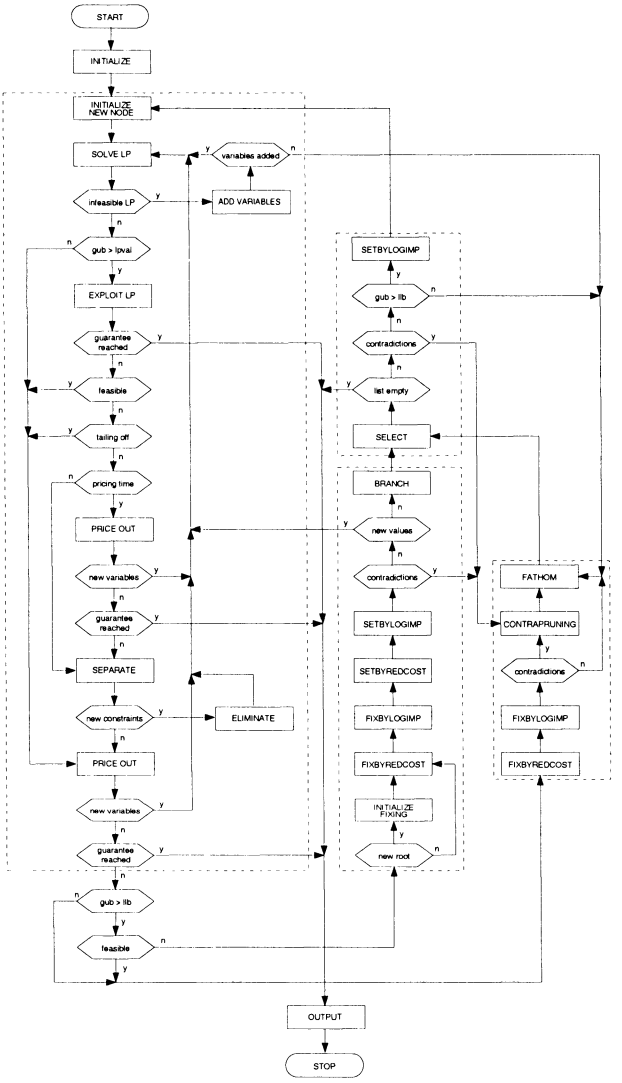

Figure 2.42. Flowchart of a Branch-and-Cut algorithm

initial sparse graph. One can use the graph of the k nearest neighbors, that is for each node we keep the edges that link this node to its k nearest neighbors. A good choice of k is 3. The resulting graph may not be Hamiltonian, not even connected if one has clusters of nodes. One therefore adds the edges of the tour yielding the initial upper bound (see further down). Another way, in the case of Euclidean instances, is to use edges of the Delaunay triangulation. This also may not be Hamiltonian, therefore one must add the edges of a tour. Helsgaun [446] has noted

that the k nearest neighbors is not in general a good choice since some instances have optimal solutions containing edges linking some nodes to far neighbors. For example he points out that in the optimal solution to att532 there is an edge that links a node to its 22nd nearest neighbor. Helsgaun also noted that an optimal tour contains between 70% and 80% of the edges of the minimum cost 1−tree. A minimum cost 1−tree is obtained by taking a minimum cost spanning tree of $G \setminus \{1\}$ to which one adds the two least cost edges incident to node 1. The *marginal value* μ_e of an edge e not in the minimum cost 1−tree T, is defined as the minimum cost of a 1−tree containing edge e minus the cost of T. A marginal cost of zero means that there exists a minimum cost 1−tree that contains e, else this cost is positive. A good choice of the sparse graph is to take all edges of T and all the edges with marginal value less than a given value α. Helsgaun shows in [446] how this can be done efficiently.

Of course we have no guarantee that the optimal solution is among those edges. Therefore one must regularly perform what is called a *pricing*, complete or partial. If one started with the k nearest neighbor graph, it seems quite logical to first search among the edges in the $k+t$ nearest neighbor graph, with t small (e.g., t=2 or 4). The edges of the distance 2 closure of the Delaunay triangulation, that is the edges linking nodes at distance 2 on the Delaunay triangulation, seems also a good choice in the other case. These edges are often referred to as those of the *reserve graph*. We will see that this pricing operation, in some sense, guides many decisions in the design of the Branch-and-Cut code. The pricing is done using the optimal solution to the dual problem. How often should it be performed is not an easy question to answer. For separation heuristics based on template paradigms, too many edges seem to be a handicap because the lp solutions seem to become too fractional. For example starting directly with the five nearest neighbor graph (instead of the three), very often yields a worse running time and more branch and cut nodes in the enumeration tree. Conversely, it happens that trying to avoid a complete pricing, one will spend quite some time separating while the lower bound is over the (yet unknown) optimal value. In the JNRT code, a reserve pricing is performed every 10 iterations, a complete pricing is performed if less than 2 variables were priced in from the reserve graph. A complete pricing is necessary in all cases when the solution to the lp is feasible (integral), or when the lower bounds exceeds the upper bound. Even in this case, only a restricted number of variables is generated before one reoptimizes. In all cases of the TSPLIB only a very small percentage of the variables are used.

Variable fixing is an important point in Branch-and-Cut. A variable can be fixed to 0 (resp. 1) if one does not cut off all optimal solutions when we never (resp. always) take that edge. This can be done in various ways. The first is by *reduced cost*. Let GUB (Global Upper Bound) be the best known value of a tour, LB (Lower Bound) be the optimal value of our current lp and assume that a complete pricing has shown that the current lp solution $x^\star$ is globally optimal. Let e be such that either $x_e^\star = 0$ or x_e is not a variable of the lp. Let $\overline{c}_e$ be its reduced cost and assume $\overline{c}_e > 0$. If $\overline{c}_e + LB \geq GUB$ on can *fix* the corresponding variable to zero, delete it from the lp, if necessary, and definitively forget about it. One advantage is that pricing will not have to be performed on it again. Note that the case "=" in the preceding formula may eliminate all optimal solutions, but in this case it is not important since one already has at hand such an optimal solution, namely the solution that yields the current GUB. With the same argument one can fix a variable x_e to 1 which is such that $x_e^\star = 1$ and $LB - \overline{c}_e \geq GUB$ (this variable is non basic and at its upper bound value).

Variables can be fixed also by *logical implication*. For example, if two incident edges have been fixed to 1, then all other edges incident to the common node can be fixed to 0. If all edges incident to a node except two are fixed to 0, then the two remaining ones can be fixed to 1. The edge linking the extremities of a path of edges fixed to 1 can be fixed to 0. These variable fixings can be done at any node of the search tree, but are only valid in the corresponding subtree. One usually talks of variable *setting* in case the node is not the *current* rootnode of the search tree, where by *current* rootnode we mean the lowest node in the search tree which is a common ancestor of all active nodes.

19.2. Constraints

As one would expect from the name of the method, constraint generation is the key point in a Branch-and-Cut code. As seen in the previous sections there are a variety of different constraint types. Should one give priority to some over the others? The strategies of the two latest codes devoted to the TSP diverge quite a bit in the separation strategies. The JNRT code will first search for violated subtour inequalities and only search for other violated inequalities once no subtour inequality is violated. It was generally taught that it was too costly to do so. The computational results that will be reported in the next section show that it is not the case. The ABCC code does not do so, following the strategy of all the preceding codes of Hong and Padberg, Grötschel and Holland, and Padberg and Rinaldi. The JNRT code only relies on

template paradigms, the ABCC code also uses more general separation procedures.

There are many theoretical ways of measuring the strength of a facet inducing inequality (see [618]).

- The number of tours that are on the facet.
- The volume of the subtour elimination polytope it cuts off.
- Its distance to the closest vertex of the subtour elimination polytope it cuts off.

Practically, the only measure that counts is the help in solving an instance? In this respect subtour elimination inequalities and comb inequalities are the strongest inequalities. For the other inequalities it depends a lot on the instance. Path inequalities seem to be the next useful family.

Before the work of Applegate, Bixby, Chvátal and Cook, several groups have thought of using general cutting planes for integer programming such as the Gomory cuts or the lift and project cuts of Balas, Ceria and Cornuéjols [68] and [69]. The major difficulty with this approach deals with the management of the variables. Remember that we are working with a sparse graph, that is not all variables are considered at a given time, and that one must be able to do pricing efficiently. To do so one must be able to retrieve very easily the coefficient of any variable in a given inequality without storing it explicitly. This can easily be done if the inequality is in closed-set form: one stores each set that defines the inequality, together with its coefficient, then the coefficient of an edge is just the weighted sum of the coboundaries it belongs to. This is why the ABCC code only keeps closed-set form inequalities in its general separation procedure.

Linear programs tend to grow very big, therefore one has to remove regularly those constraints that are no longer active. A good idea is also to remove those that have been very little active in the past k iterations, that is have had an optimal associated dual variable close to zero. This has to be done with caution in order not to destroy the basis. Note that we never delete variables except when fixing to 0. This may be necessary for some large instances in which variables would flow in. Removed subtour elimination inequalities are discarded since they can easily be recovered, the others are put into a pool which is regularly searched through. A too costly strategy for large and difficult instances and which has been in use in the first TSP codes, is to search first the pool for violated inequalities, and then call the separation heuristics in case of failure. This has the advantage of guaranteeing that no inequality

is duplicated in the pool. This strategy is no longer used because the pool of constraints grows very big and searching it is very time consuming. One must then make sure not to store the same inequality twice.

19.3. Upperbounding

We now address the problem of finding a good tour. The aim is on one hand to enable efficient variable fixing, and on the other not to explore useless branches of the search tree. Moreover, a very good upper bound makes the choice of the tree exploration strategy less critical.

Traditionally a heuristic is called to yield a first value and a tour, the edges of which are used in the sparse graph. Most codes try to exploit the lp solution, i.e., they try to take advantage of the information the lp solution $x^\star$ carries. It seems more probable that an edge e with $x^\star$ close to 1 is in an optimal solution than one with such a value close to 0. Based on this, an initial tour is built and a Lin-Kernighan ([563],[446]) type heuristic is called to try and improve it. It is important not to try and improve twice a same tour. This can be done using a hash function, see [475], [446] and [587].

Any integer feasible solution of the lp encountered on the way also can yield a better tour.

19.4. Branching

Branching occurs when one has renounced to solve the instance with the pure cutting plane method. This may occur either because one does not find any more violated inequalities or because the lower bound has not progressed in a significant manner in the past iterations. This last phenomenon is known as *tailing off*. Before branching a complete pricing is performed in order to be able to fix as many variables as possible, since these fixings will be definitive. The value of the objective function of the current lp, that is the lower bound, is known as the *root value* or *root bound*. We assume the reader familiar with the Branch and Bound terminology.

Since, in some sense, resorting to branching is felt as a failure, not much attention has been dedicated for a long time to this operation. Unfortunately powers of 2 grow very fast and it has become obvious that one must do that operation carefully.

The objective of branching is to split a certain subset of tours into two smaller subsets, to each subset corresponds a lp relaxation which does not contain the current fractional solution as a feasible solution. This can be done in various ways. The classical way is to choose some edge e such that the corresponding variable $x_e^\star$ is fractional. The tours

are split into those that contain e and those that do not. This is done by modifying the bounds of the corresponding variable in the current lp, moving the lower-bound to 1 in one case, the upper-bound to 0 in the other. This creates two new problems that are added to the list of problems to solve. To choose among all fractional variables, the mostly used criterion is to choose one of value close .5, and among those one with highest cost. Clochard and Naddef [203] advocate using subtour inequalities to perform branching. Let $S \subset V$, with $x^\star(\delta(S))$ not too close to an integer value, say $2k < x^\star(\delta(S)) < 2k + 2$. Then one can split the tours into those that cross the boundary of S at most $2k$ times and those that do so at least $2k + 2$ times. This is done by adding the appropriate inequality to the current lp. The difficulty is in finding a convenient set S. In the JNRT code this is done by using handles of previously found combs, star or clique trees, but there is a lot of room for improvement in this choice.

The branching inequality could be anything else. Another possibility is, given two disjoint sets $S, T \subset V$, split the tours between those which use at least an edge from $(S : T)$ and those which don't. This is efficient if the shortest edge of $(S : T)$, in terms of cost (not $x^\star$ value) is large and a fractional amount of these edges are used in the solution $x^\star$, an amount close to .5 also seems a good choice. Figure 2.43 illustrates this last branching rule on a fractional solution to fl1400. We have not put in the edge values. Each cluster of nodes A, B and C contains several hundreds of nodes. We have $x^\star(A : B) = x^\star(A : C) = x^\star(B : C) = 0.5$. There is only one edge e between B and C with non-zero $x^\star$ value, and it is likely to be selected by the standard criterion (close to .5 and high cost). What will happen on the side of the search tree in which one will have set $x_e = 0$? Since many edges linking B and C have costs very close to that of e, it is very likely that one will appear with that same value (or several adding up to that value). The result will be a problem that will have a lower bound almost equal to that of its father and no progress will have been made. This is typical of a case in which one should branch by saying that $x(B : C) = 0$ on one side and $x(B : C) \geq 1$ on the other side. Two other candidates are of course $x(A : C)$ and $x(A : B)$. Note that this fractional point contains some maximally violated combs which most certainly could be found using the domino separation procedure, but not by the heuristics of Naddef and Thienel, which is certainly why some codes such as the ABCC code do very well on this instance and the JNRT code does very poorly. One of these combs is shown in Figure 2.43, the set C which is one of the teeth, the other sets defining the comb are shown in thin lines.

In general branching on variables does pretty well, but it is disastrous on difficult instances. Trying to perform better on these instances unfortunately has a price on the easier instances. But this is not specific to the TSP. In fact, NP-hardness means that there are very difficult instances which may be rare. One has at some point to decide how much one is willing to sacrifice on the solution time for the easy instances in order to perform not too badly on those rare instances.

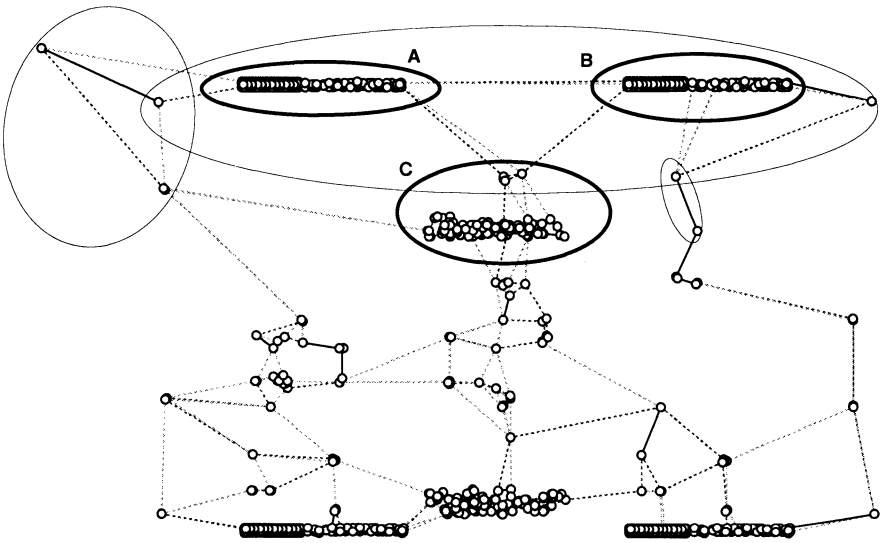

Figure 2.43. Illustration for the third branching rule

Applegate et al. introduced *strong* branching in an effort to optimize the probability of performing a good choice. It roughly goes as follows: choose a large enough set of candidates for branching (variables or inequalities). Test each of them by solving the corresponding lp relaxation only. The standard criterion is to choose the candidate for which the smallest of the two lower bounds of its sons, is the largest. Jünger et al. adapted this criterion a little by choosing among those with the same rounded smallest lower bound, the one with largest other lower bound, that is between a candidate yielding two bounds of 125.85 and 125.99 and one with 125.45 and 127.36, they choose the second. There is no reason, other than simplicity, to restrict to the rounded value; a better choice would be to choose among those which are within a certain interval, the one which has a larger upperbound. Computational experiments show that the time spent in choosing a good candidate definitively pays off, at least on difficult instances.

While performing this strong branching, one may encounter a bound which exceeds the upper bound. One has to take advantage of this by either setting the corresponding variable to the other value, or by adding the converse inequality to the lp.

We are left with the problem of choosing which open problem to solve next. It is well known in Branch-and-Bound that if the initial upper-bound equals the optimal value, whatever exploration strategy one uses, the same tree will be explored. This is no longer true in Branch-and-Cut. This is due to the fact that the inequalities one generates are globally valid, that is wherever they have been generated in the search tree, they are valid at any node of the search tree. The non-tivial inequalities, as already mentioned, are stored in a pool through which one searches regularly. The content of that pool depends on the previously visited nodes of the search tree, and since Branch-and-Cut is highly unrobust, the whole search may be considerably changed. Nevertheless it seems a good idea to go depth first if one believes that the current feasible solution is optimal. If not it seems that branching on the most promising open problem (the one with lowest lower bound) is the best choice.

20. Computational results

The two latest implementations of the Branch-and-Cut method for the TSP are that of Applegate, Bixby, Chvátal and Cook (ABCC) known as *CONCORDE*, and that of Jünger, Naddef, Reinelt and Thienel (JNRT). The latter relies on a general purpose Branch-and-Cut system known as *ABACUS* and uses the separation heuristics of Padberg and Rinaldi for subtours, and the ones of Naddef and Thienel for the other inequalities. The initial upperbounding is done using Andre Rohe's TSP implementation of Lin Kernighan. *ABACUS* is a system that manages all parts of Branch-and-Cut which are problem independent, making the development of optimization software based on Branch-and-Cut much easier since one may concentrate on problem-specific technicalities. Of course there is a price, which for the TSP ranges somewhere between 10% and 25% on CPU time.

Another main difference between these two codes is the separation strategies. JNRT only calls for separation of violated inequalities which are not subtour elimination inequalities when none of the latter are violated. They also restrict to separation into known classes of facet inducing inequalities. This is not the case for ABCC, as already seen.

Table 20 gives a comparison of times and number of nodes of the search tree between the two codes. The times reported below are for JNRT on a Sun Ultra-60 with a 295 MHz processor. For CONCORDE (ABCC) on

a 500 MHz Compaq XP1000 workstation. Following some benchmarking of ABCC, and due to the fact that JNRT use a general purpose Branch-and-Cut code, a factor of 4 or 5 seems to be reasonable to compare CPU times. These times, like all other figures should be taken with caution. Branch-and-Cut is a highly unrobust method: a slight change in some parameter setting may change considerably the number of nodes of the Branch-and-Cut tree and the CPU time. Times also fluctuate considerably depending on whether or not the upper-bound has been found early enough to save useless computations. For the JNRT code we also give, for some relevant instances, the results of runs in which the optimum value was entered as the initial upper bound. The interest of such data is to differentiate those instances for which the separation of valid inequalities is the difficulty from those for which it is the upperbounding. The data for the instance fnl4461 for the ABCC code also corresponds to the initial upperbound set to the optimal value and for pcb3038, in the ABCC code the initial tour happens to be the optimal one. JNRT chose the option to privilege separation over branching, in the aim of testing how good the separation heuristics are. It may have been faster sometimes to branch much earlier.

A first conclusion is that separation using template paradigms does pretty well in most cases. The local cut separation procedures helps in some difficult instances. Another conclusion is that the size of a problem is not really what makes it difficult. Finally, if one looks into the recent history of TSP solving, one realizes that good separation is the first clue to success, the second being clever branching. The instance ts225 was solved by Applegate et al. for the first time in over 5000 BC nodes in about two years of CPU time, then in a little over an hour and about 40 BC nodes by Naddef and Thienel using their separation routines, and now is solved in only a few seconds by CONCORDE. The same is illustrated by the instances pcb3038 and fnl4461 which were first solved in a few years of CPU time and now in less than one day of CPU. Table 2.2 gives the results of CONCORDE on very large instances. The parameter settings are in general different from the ones used for the previous table. Sometimes the optimal tour value was an input. See the web page of CONCORDE for more information.

21. Conclusion

In this chapter we have seen an example of the efforts that must be carried out in order to achieve good performance in the search for optimal solutions of large instances of the symmetric TSP via Branch-and-Cut. This effort is two-fold. First a deep theoretical research in

	BC nodes			time in mn:ss		
Name	JNRTopt	JNRT	ABCC	JNRTopt	JNRT	ABCC
pr76	-	1	1	-	0:35	0:02
gr120	-	1	1	-	0:01	0:02
pr136	-	1	1	-	0:02	0:04
pr144	-	1	1	-	0:01	0:02
pr152	-	1	1	-	0:02	0:08
u159	-	1	1	-	0:01	0:01
rat195	-	1	1	-	1:41	0:22
d198	-	3	3	-	0:20	0:12
kroA200	-	1	1	-	0:18	0:07
kroB200	-	1	1	-	0:11	0:04
gr202	-	1	1	-	0:09	0:05
ts225	-	57	1	-	86:26	0:21
pr226	-	1	1	-	0:03	0:04
gr229	-	3	3	-	3:31	0:39
gil262	-	1	1	-	0:43	0:13
pr264	-	1	1	-	0:03	0:03
pr299	-	1	3	-	1:06	0:17
lin318	-	1	1	-	0:40	0:10
rd400	13	15	15	2:39	4:38	2:28
fl417	-	1	5	-	0:55	0:58
gr431	9	9	13	7:02	10:40	2:13
pr439	-	5	15	-	8:39	3:36
pcb442	-	9	9	-	1:33	0:50
d493	-	1	5	-	10:29	1:53
att532	5	5	7	6:23	9:47	1:50
ali535	-	1	3	-	2:18	0:53
pa561	7	9	17	27:16	43:20	4:07
u574	-	1	1	-	0:35	0:23
rat575	7	11	25	4:47	20:38	6:03
p654	-	1	3	-	0:15	0:15
gr666	-	1	3	-	8:18	0:50
u724	9	9	11	10.05	20:45	3:45
rat783	-	1	1	-	2:13	0:38
dsj1000	-	1	7	-	75:27	6:50
pr1002	-	1	1	-	0:59	0:34
u1060	7	29	21	9:11	55:09	9:31
vm1084	7	7	11	66:25	81:21	10:04
pcb1173	13	25	19	21:49	77:23	7:48
rl1304	-	1	1	-	12:01	3:09
rl1323	33	55	25	327:35	407:01	62:20
nrw1379	-	7	19	-	29:09	9:38
d1655	-	1	5	-	25:33	4:23
vm1748	9	19	17	155:56	318:57	37:03
pr2392	-	1	1	-	7:44	1:47
pcb3038	265	-	313	5771mn	-	1347mn
fnl4461	409	-	213	9410mn	-	890mn

Table 2.2. Computational results

Name	BC nodes	time
rl5915	161	644h 20mn
rl5934	205	163h 35mn
pla7397	101	119h 10mn
rl11849	431	$\simeq$ 155 days
usa13509	9539	$\simeq$ 4 years
d15112	164569	$\simeq$ 22.6 years

Table 2.3. Very large instances with Concorde

order to study the convex hull of the solutions. Note that even if one uses separation routine outside the template paradigm, this study is necessary, since some steps of the algorithm need these theoretical results to justify them. Following that theoretical study, one must invest in a relevant algorithmic and implementation effort. The implementation effort is unfortunately now far too high for a newcomer. If this very exciting field is not to die, the only hope is that large pieces of existing source code be made available to anybody interested in developing a new challenging program. We hope that this will be the case in a near future.